MQSeries® for OS/390®

# System Setup Guide

*Version 5 Release 2*

SC34-5651-00

**Note!**

Before using this information and the product it supports, be sure to read the general information under "Appendix D. Notices" on page 247.

**First edition (November 2000)**

This edition applies to MQSeries for OS/390 Version 5 Release 2, and to all subsequent releases and modifications until otherwise indicated in new editions.

This book is based on parts of the *MQSeries for OS/390 System Management Guide*, SC34-5374-01.

# Contents

# Figures

# Tables

# About this book

This book describes the tasks that you have to perform to customize MQSeries® after you have installed it. It also describes how to monitor system use and performance, and how to set up security.

This book is based on parts of the *MQSeries for OS/390 Version 2.1 System Management Guide*.

The *System Management Guide* has been replaced by the following three books:

- *MQSeries for OS/390® Concepts and Planning Guide*. This book describes the concepts of MQSeries for OS/390; it does not describe the concepts of MQSeries messaging and queueing. If you are not familiar with these concepts you should read *MQSeries: An Introduction to Messaging and Queuing*. It also describes how to plan your MQSeries for OS/390 systems.
- *MQSeries for OS/390 System Setup Guide*. (This book.)
- *MQSeries for OS/390 System Administration Guide*. This book describes how to operate MQSeries, how to perform recovery and restart tasks, and how to use the panel interface and utilities supplied with MQSeries.

Changes to the information in this book since the last edition of the *System Management Guide* are marked with vertical bars in the left-hand margin.

## Who this book is for

This book is intended for system programmers, system administrators, and security administrators.

## What you need to know to understand this book

This book assumes you are familiar with the basic concepts of:

- CICS®
- DB2® (if you intend to use queue-sharing groups)
- IMS™
- OS/390 Coupling Facility (if you intend to use queue-sharing groups)
- OS/390 job control language (JCL)
- OS/390 Time Sharing Option (TSO)

## Conventions used in this book

- Throughout this book, the term *object* refers to any MQSeries queue manager, queue, namelist, channel, storage class, or process.
- The examples in this book are taken from a queue manager with a command prefix string (CPF) of +CSQ1. The commands are shown in UPPERCASE.
- CICS means both CICS Transaction Server for OS/390 and CICS for MVS/ESA unless otherwise stated. IMS means IMS/ESA® unless otherwise stated.
- Throughout this book, the default value `thlqual` is used to indicate the target library high-level qualifier for MQSeries data sets in your installation.
- Throughout this book, the term *distributed queuing* refers to the distributed queuing feature (also known as the “non-CICS mover”). The term *distributed*

*queuing using CICS ISC* is used to refer to the optional CICS distributed queuing feature (also known as the “CICS mover”).

## What’s new for this release

This section describes the new function that has been added for this release of MQSeries for OS/390.

### Queue-sharing groups

- You can group your queue managers into a queue-sharing group. These queue managers can access the same set of shared queues. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- A new system parameter (QSGDATA) sets the queue-sharing group parameters for your queue manager. This is described in “Using CSQ6SYSP” on page 31.
- You can now define an object once on one queue manager and then use the object definition on other queue managers in the queue-sharing group. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now send commands to all queue managers in a queue-sharing group by issuing the command on one member of the group. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now use intra-group queuing to send messages between queue managers in a queue-sharing group, without setting up channels. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.

### Channel initiator

- You can now use shared channels. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.
- You can now stop a receiver channel automatically and start a new one in its place when a request to start a duplicate receiver channel is received by using the ADOPTMCA and ADOPTCHK parameters of the CSQ6CHIP channel initiator parameter macro. This is described in the *MQSeries Intercommunication* manual; CSQ6CHIP is described in “Using CSQ6CHIP” on page 51. (This function was available as a PTF for previous releases.)
- You can now specify a range of port numbers to be used when binding outbound channels by using the OPORTMIN and OPORTMAX parameters of the CSQ6CHIP channel initiator parameter macro. This is described in “Using CSQ6CHIP” on page 51. (This function was available as a PTF for previous releases.)
- You can now specify a single IP address upon which the TCP/IP listener accepts inbound connection requests. This is described in the *MQSeries MQSC Command Reference*.
- You can now specify that a TCP listener should register with Workload Manager for Dynamic Domain Name Services by using the DNSWLM and DNSGROUP parameters of the CSQ6CHIP channel initiator parameter macro. This is described in “Using CSQ6CHIP” on page 51.
- You can now specify an LU name that is defined as part of a generic resource for an LU 6.2 listener by using the LUGROUP parameter of the CSQ6CHIP channel initiator parameter macro. This is described in “Using CSQ6CHIP” on page 51.

## Commands

- The following commands have been added:

| | |
|---|---|
| **CLEAR QLOCAL** | Clear messages from a local queue. |
| **DISPLAY GROUP** | Display information about the queue-sharing group to which the queue manager is connected. |
| **DISPLAY QSTATUS** | Display queue status information. |
| **MOVE QLOCAL** | Move messages from one queue to another. |
| **RESET QSTATS** | Display information about how many messages have been put to and retrieved from a queue. |

  These commands are described in the *MQSeries MQSC Command Reference*.
- The QSGDISP attribute has been added to many commands to specify the object disposition (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The CHLDISP attribute has been added to the channel commands to specify the channel disposition (described in the *MQSeries MQSC Command Reference*).
- The CMDSCOPE attribute has been added to most commands to specify the command scope (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The responses to many commands have changed.

## System parameters

- The following new system parameters have been added (they are all described elsewhere in this chapter):

| | |
|---|---|
| **CSQ6SYSP** | QSGDATA, RESAUDIT |
| **CSQ6LOGP** | DEALLCT, MAXRTU |
| **CSQ6ARVP** | UNIT2 |

- The default settings for the following system parameters have changed:

| | |
|---|---|
| **CSQ6SYSP** | The default for LOGLOAD is now 500 000. |
| **CSQ6LOGP** | The default for INBUFF is now 60 KB and the default for OUTBUFF is now 4 000 KB. |

  These are described in “Task 16: Tailor your system parameter module” on page 30.
- The MAXALLC parameter of CSQ6LOGP is no longer used.

## System object samples

- A new system object sample (CSQ4INSS) that defines the objects required for queue-sharing groups has been added (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The default buffer pool, storage class and page set definitions have been changed. These are explained in the CSQ4INP1 and CSQ4INYG sample data sets (described in the *MQSeries for OS/390 Concepts and Planning Guide*).
- The sample object data set for the basic IVP has been renamed to CSQ4IVPQ, and a new sample object data set called CSQ4IVPG has been added for the queue-sharing group IVP.

- The default region sizes have been increased to 0M for the queue manager and channel initiator address spaces.

## Logs

- The default settings for log placement and size have been changed. These changes are described in the CSQ4BSDS sample data set.
- You can now specify that you want both copies of the archive log to be written to different device types by using the UNIT and UNIT2 parameters of the CSQ6ARVP system parameter macro. This is described in “Using CSQ6ARVP” on page 42.
- You can now restart the log archive process after a failure by using the ARCHIVE LOG CANCEL OFFLOAD command. This command is described in the *MQSeries MQSC Command Reference*.
- You can now optimize archive log reading from tape devices using the MAXRTU and DEALLCT parameters of the CSQ6LOGP system parameter macro. You can display and reset these parameters using the DISPLAY LOG and SET LOG commands. CSQ6LOGP is described in “Using CSQ6LOGP” on page 38; the commands are described in the *MQSeries MQSC Command Reference*.

## Security

- You can now use one set of security profiles to control security for all the queue managers in a queue-sharing group. This is described in “Profiles to control queue-sharing group or queue manager level security” on page 141.
- You can now control whether RACF® audit records are written for RESLEVEL security checks using the RESAUDIT parameter of the CSQ6SYSP system parameter module. This is described in “Using CSQ6SYSP” on page 31.

## Statistics and accounting

- You can now use the SMF data collection broadcast to gather MQSeries statistics and accounting records. This is described in “Using System Management Facility” on page 110.
- You can now gather performance statistics for the Coupling Facility manager and DB2 manager. This is described in “Chapter 10. Interpreting MQSeries performance statistics” on page 115.
- You can now gather accounting data at queue and thread level. This is described in “Chapter 11. Interpreting MQSeries accounting data” on page 127.

## Operations and control panels

- The operations and control panels have been changed extensively to include the new function. For example, you are now asked to enter the disposition of objects that you are working with, and this information is included when you use the panels to display an object. This is described in the *MQSeries for OS/390 System Administration Guide* and in the help supplied with the operations and control panels.
- The DEFINE action has been changed to DEFINE LIKE.
- You can now get and collate information for the whole queue-sharing group.
- You can now display queue status information.

## Dead-letter queue

There is a new utility program (CSQUDLQH) which processes messages on the dead-letter queue. This is described in the *MQSeries for OS/390 System Administration Guide*.

## Application programming

- MQSeries messages can now be up to 100 MB in length. (This function was available as a PTF for previous releases.)
- The Application Messaging Interface (AMI) provides a simple interface to the Message Queue Interface (MQI). Application programmers can use this interface to write applications without needing to understand all the functions available in the MQI. The functions that are required in a particular installation are defined by a system administrator. This function is described in the *MQSeries Application Messaging Interface* manual.
- The MQRC_STORAGE_MEDIUM_FULL return code has been added. This is described in the *MQSeries Application Programming Reference*.
- New code pages have been added, including one for UNICODE. These are described in the *MQSeries Application Programming Reference*.
- You can now specify options when you connect to a queue manager using the **MQCONNX** call. This is described in the *MQSeries Application Programming Reference*.

## What’s new for this release

# Part 1. Customizing your queue managers

# Chapter 1. Preparing for customization

The *MQSeries for OS/390 Program Directory* lists the contents of the MQSeries installation tape, the program and service level information for MQSeries, and describes how to install MQSeries under OS/390 using the System Modification Program Extended (SMP/E).

When you have installed MQSeries, you must carry out a number of tasks before you can make it available to users. Refer to the following sections for a description of these tasks:
- "Chapter 2. Customizing your queue managers" on page 7
- "Chapter 4. Testing your queue manager" on page 73
- "Part 5. Setting up security" on page 135

If you are migrating from a previous version of MQSeries for OS/390, you don't need to perform most of the customization tasks. Refer to "Chapter 3. Migrating your queue manager from a previous version" on page 63 for information about the tasks you have to perform.

## Installable features

MQSeries for OS/390 comprises the following features:

**Base** This is required; it comprises all the main functions, and includes the distributed queuing facility (supporting both TCP/IP and APPC communications).

**National language features**
These contain error messages and panels in all the supported national languages. Each language has a language letter associated with it. The languages and letters are:

| | |
|---|---|
| **C** | Simplified Chinese |
| **E** | U.S. English (mixed case) |
| **K** | Japanese |
| **U** | U.S. English (uppercase) |

You must install at least one of these (you can install more than one).

| **CICS Mover feature**
This is optional; it is required only if you are using CICS ISC for distributed queuing.

**Client Attachment feature**
This is optional; it is only required if you are going to attach clients to your MQSeries for OS/390 subsystem. When you have installed this feature, there are no configuration parameters to set before you can attach clients to MQSeries for OS/390. Administration for clients is available even if you don't install this feature.

**Internet Gateway feature**
This is optional; it is only required if you want to use the MQSeries Internet Gateway. This is described in the *MQSeries Internet Gateway User's Guide*. (This online book is supplied with the Internet Gateway.)

## Libraries that exist after installation

MQSeries is supplied with a number of separate load libraries. Table 1 shows the libraries that might exist after you have installed MQSeries.

*Table 1. MQSeries libraries that exist after installation*

| **Name** | **Description** |
|---|---|
| thlqual.SCSQANLC | Contains the load modules for the Simplified Chinese version of MQSeries. |
| thlqual.SCSQANLE | Contains the load modules for the U.S. English (mixed case) version of MQSeries. |
| thlqual.SCSQANLK | Contains the load modules for the Japanese version of MQSeries. |
| thlqual.SCSQANLU | Contains the load modules for the U.S. English (uppercase) version of MQSeries. |
| thlqual.SCSQASMS | Contains source for assembler sample programs. |
| thlqual.SCSQAUTH | The main repository for all MQSeries product load modules; it also contains the default parameter modules, CSQZPARM and CSQXPARM. This library must be APF-authorized. |
| thlqual.SCSQCICS | Contains the load modules that must be included in the CICS DFHRPL concatenation. These are separated from the main MQSeries load library so that the number of modules in the concatenation search is kept to a minimum to improve performance and to avoid the need for APF authorization. |
| thlqual.SCSQCLST | Contains CLISTs used by the mail manager sample program. |
| thlqual.SCSQCOBC | Contains COBOL copybooks, including copybooks required for the sample programs. |
| thlqual.SCSQCOBS | Contains source for COBOL sample programs. |
| thlqual.SCSQCPPS | Contains C source for C++. |
| thlqual.SCSQC37S | Contains source for C/370™ sample programs. |
| thlqual.SCSQC370 | Contains C/370 headers, including headers required for the sample programs. |
| thlqual.SCSQDEFS | Contains side definitions for C++ and the DB2 DBRMs for shared queuing. |
| thlqual.SCSQEXEC | Contains REXX execs to be included in the SYSEXEC or SYSPROC concatenation if you are using the MQSeries operations and control panels. |
| thlqual.SCSQHPPS | Contains header files for C++. |
| thlqual.SCSQINST | Contains JCL for installation jobs. |
| thlqual.SCSQLINK | Early code library. Contains the load modules that must be in the link list because they are loaded at system initial program load (IPL). The library must be APF-authorized and must be in the link list. |
| thlqual.SCSQLOAD | Load library. Contains load modules for non-APF code, user exits, utilities, samples, installation verification programs, and adapter stubs. The library does not need to be APF-authorized and does not need to be in the link list. |
| thlqual.SCSQMACS | Contains Assembler macros including: sample macros, product macros, and system parameter macros. |
| thlqual.SCSQMAPS | Contains CICS mapsets used by sample programs. |
| thlqual.SCSQMSGC | Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the Simplified Chinese language feature for the MQSeries operations and control panels. |
| thlqual.SCSQMSGE | Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the U.S. English (mixed case) language feature for the MQSeries operations and control panels. |
| thlqual.SCSQMSGK | Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the Japanese language feature for the MQSeries operations and control panels. |
| thlqual.SCSQMSGU | Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the U.S. English (uppercase) language feature for the MQSeries operations and control panels. |
| thlqual.SCSQMVR1 | Contains the load modules for distributed queuing when using TCP/IP with the OpenEdition® sockets or IUCV interface, or LU 6.2. This library must be APF-authorized. |

*Table 1. MQSeries libraries that exist after installation (continued)*

| Name | Description |
|---|---|
| thlqual.SCSQMVR2 | Contains the load modules for distributed queuing when using TCP/IP with the SOLVE:TCPaccess interface, or LU 6.2. This library must be APF-authorized. |
| thlqual.SCSQPLIC | Contains PL/I headers. |
| thlqual.SCSQPLIS | Contains source for PL/I sample programs. |
| thlqual.SCSQPNLA | Contains IPCS panels, for the dump formatter, to be included in the ISPPLIB concatenation. Also contains panels for MQSeries sample programs. |
| thlqual.SCSQPNLC | Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the Simplified Chinese language feature for the MQSeries operations and control panels. |
| thlqual.SCSQPNLE | Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the U.S. English (mixed case) language feature for the MQSeries operations and control panels. |
| thlqual.SCSQPNLK | Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the Japanese language feature for the MQSeries operations and control panels. |
| thlqual.SCSQPNLU | Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the U.S. English (uppercase) language feature for the MQSeries operations and control panels. |
| thlqual.SCSQPROC | Contains sample JCL and default system initialization data sets. |
| thlqual.SCSQSKL | Contains ISPF skeletons to be included in the ISPSLIB concatenation if you are using the MQSeries operations and control panels. |
| thlqual.SCSQSNLC | Contains the load modules for the Simplified Chinese versions of the MQSeries modules that are required for special purpose function (for example the early code). |
| thlqual.SCSQSNLE | Contains the load modules for the U.S. English (mixed case) versions of the modules that are required for special purpose function (for example the early code). |
| thlqual.SCSQSNLK | Contains the load modules for the Japanese versions of the MQSeries modules that are required for special purpose function (for example the early code). |
| thlqual.SCSQSNLU | Contains the load modules for the U.S. English (uppercase) versions of the MQSeries modules that are required for special purpose function (for example the early code). |
| thlqual.SCSQTBLC | Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the Simplified Chinese language feature for the MQSeries operations and control panels. |
| thlqual.SCSQTBLE | Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the U.S. English (mixed case) language feature for the MQSeries operations and control panels. |
| thlqual.SCSQTBLK | Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the Japanese language feature for the MQSeries operations and control panels. |
| thlqual.SCSQTBLU | Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the U.S. English (uppercase) language feature for the MQSeries operations and control panels. |
| **Attention**: Do *not* modify or customize any of these libraries. If you want to make changes, copy the libraries, and make your changes to the copies. | |

## Libraries that exist after installation

# Chapter 2. Customizing your queue managers

This chapter leads you through the various stages of customizing MQSeries after you have successfully installed it. The installation process is described in the *MQSeries for OS/390 Program Directory*.

Samples are supplied with MQSeries to help you with your customization. The sample data set members have names beginning with the four characters CSQ4 and are in the library thlqual.SCSQPROC.

## Before you start

Before you perform the customization tasks in this chapter, there are a number of configuration options that you should consider because they affect the performance of MQSeries for OS/390. These options are discussed in the *MQSeries for OS/390 Concepts and Planning Guide*.

For each task you must also consider:

1. ***Whether the task must be repeated for each MQSeries subsystem.***

   Some of these tasks you need only do once, regardless of the number of MQSeries subsystems, while others must be repeated for each MQSeries subsystem. Each task description tells you which category that task belongs to.
2. ***Whether the task requires an IPL.***

   Some tasks might only take effect following an OS/390 system initial program load (IPL). For example, an IPL might be required:
   - By any task that changes certain OS/390 system parameters
   - When you change certain tables used by an external security manager, such as RACF

   Therefore, make sure you have completed **all** the necessary tasks before you IPL the system. Each task description tells you whether an IPL is required. In general, an IPL is needed when you install and customize MQSeries, but not when you add a new MQSeries subsystem.

If you are migrating from a previous version of MQSeries for OS/390, you might not need to perform all of these tasks. Each task description tells you whether you need to perform the task when migrating.

### Identify the national language support libraries

You need to specify the appropriate national language support libraries in the JCL that you will use for running MQSeries (as described in the following sections). Each language is identified by a language letter:

| | |
|---|---|
| **C** | Simplified Chinese |
| **E** | U.S. English (mixed case) |
| **K** | Japanese |
| **U** | U.S. English (uppercase) |

Table 2 on page 8 shows the names of the libraries for the language features; the language letter is the last letter of the library name.

## Before you start

Table 2. National language feature libraries

| Description | Japanese | Simplified Chinese | U.S. English (mixed case) | U.S. English (uppercase) |
|---|---|---|---|---|
| Load modules | thlqual.SCSQANLK | thlqual.SCSQANLC | thlqual.SCSQANLE | thlqual.SCSQANLU |
| ISPF messages | thlqual.SCSQMSGK | thlqual.SCSQMSGC | thlqual.SCSQMSGE | thlqual.SCSQMSGU |
| ISPF panels | thlqual.SCSQPNLK | thlqual.SCSQPNLC | thlqual.SCSQPNLE | thlqual.SCSQPNLU |
| Special purpose function (for example, early code) | thlqual.SCSQSNLK | thlqual.SCSQSNLC | thlqual.SCSQSNLE | thlqual.SCSQSNLU |
| ISPF tables | thlqual.SCSQTBLK | thlqual.SCSQTBLC | thlqual.SCSQTBLE | thlqual.SCSQTBLU |

## Customization summary

The following table lists all the steps required to customize MQSeries for OS/390. It also indicates the following:

- Whether the step has to be performed once only, or repeated for each queue manager.
- Whether the step requires an IPL.
- Whether the step is required if you are migrating from a previous version of MQSeries. Some steps might be needed, depending on what you decide about data set and queue manager names; these are marked 'Review'.

Table 3. Customization summary

| Task | Page | Requires an IPL | Required when migrating | Repeat for each queue manager |
|---|---|---|---|---|
| OS/390 customization tasks | | | | |
| Task 1: Identify the OS/390 system parameters | 10 | – | Review | – |
| Task 2: Review the number of system LXs | 11 | ✓ | Review | – |
| Task 3: Define the MQSeries subsystem to OS/390 | 12 | ✓ | – | ✓ |
| Task 4: Update the OS/390 link list and LPA | 16 | ✓ | Review | – |
| Task 5: APF authorize the MQSeries load libraries | 17 | – | Review | ✓ |
| Task 6: Update the OS/390 program properties table | 18 | ✓ | – | ✓ |
| Task 7: Create procedures for the MQSeries subsystem | 19 | – | Review | ✓ |
| Task 8: Create procedures for the channel initiator | 20 | – | Review | ✓ |
| Task 9: Set up the DB2 environment | 21 | – | ✓ | – |
| Task 10: Set up the Coupling Facility | 22 | – | ✓ | ✓ |
| Task 11: Implement your ESM security controls | 23 | ✓ | Review | ✓ |
| MQSeries customization tasks | | | | |
| Task 12: Customize the initialization input data sets | 24 | – | ✓ | ✓ |
| Task 13: Create the bootstrap and log data sets | 27 | – | – | ✓ |
| Task 14: Define your page sets | 28 | – | – | ✓ |
| Task 15: Add the MQSeries entries to the DB2 data-sharing group | 29 | – | ✓ | ✓ |
| Task 16: Tailor your system parameter module | 30 | – | ✓ | ✓ |
| Task 17: Tailor the channel initiator parameter module | 50 | – | ✓ | ✓ |

*Table 3. Customization summary (continued)*

| Task | Page | Requires an IPL | Required when migrating | Repeat for each queue manager |
|---|---|---|---|---|
| Task 18: Set up Batch, TSO, and RRS adapters | 58 | – | Review | – |
| Task 19: Set up the operations and control panels | 59 | – | Review | – |
| Task 20: Include the MQSeries dump formatting member | 61 | – | ✓ | – |
| Task 21: Suppress information messages | 62 | – | – | – |

## Task 1: Identify the OS/390 system parameters

- *You need only perform this task once.*
- *You might need to perform this task when migrating from a previous version.*

Some of the tasks involve updating the OS/390 system parameters. You need to know which ones were specified when the system IPL was performed. SYS1.PARMLIB(IEASYSpp) contains a list of parameters that point to other members of SYS1.PARMLIB (where pp represents the OS/390 system parameter list that was used to IPL the system).

The entries you need to find are:

**For "Task 2: Review the number of system LXs" on page 11:**
NSYSLX=nn is the number of linkage indexes reserved for system LXs (member IEASYSxx)

**For "Task 3: Define the MQSeries subsystem to OS/390" on page 12:**
SSN=ss points to the defined subsystem list (member IEFSSNss)

**For "Task 4: Update the OS/390 link list and LPA" on page 16:**
LNK=kk points to the link list (member LNKLSTkk)

**For "Task 5: APF authorize the MQSeries load libraries" on page 17:**
PROG=xx or APF=aa point to the Authorized Program Facility (APF) authorized library list (member PROGxx or IEFAPFaa)

**For "Task 6: Update the OS/390 program properties table" on page 18:**
SCH=xx points to the Program Properties Table (PPT) (member SCHEDxx)

## Task 2: Review the number of system LXs

- *You need only perform this task once.*
- *You must IPL your system before changes to NSYSLX take effect.*
- *You might need to perform this task when migrating from a previous version.*

Each MQSeries subsystem defined in the subsystem name table reserves one system linkage index at IPL time. This system linkage index is reused if the MQSeries subsystem is stopped and restarted. The NSYSLX parameter in IEASYSxx defines the number of linkage indexes (in addition to those in the system function table) to be reserved as system linkages. The default number is 55.

If your environment has a number of subsystems defined that use system linkage indexes (for example, DB2, IRLM, and IMS V5), you might need to increase the value of NSYSLX when you define MQSeries subsystems. Each MQSeries subsystem reserves one system linkage index, and each instance of the distributed queuing feature reserves one non-system linkage index.

## Task 3: Define the MQSeries subsystem to OS/390

- *Repeat this task for each MQSeries subsystem.*
- *You might have to IPL the system before these changes take effect.*
- *You do not need to perform this task when migrating from a previous version.*

### Updating the subsystem name table

The subsystem name table of OS/390, which is taken initially from the SYS1.PARMLIB member IEFSSNss, contains the definitions of formally defined OS/390 subsystems. To define each MQSeries subsystem, you must add an entry to this table, either by changing the IEFSSNss member of SYS1.PARMLIB, or by using the SETSSI OS/390 command if it is available.

If you use the SETSSI command, the change takes effect immediately, otherwise you must IPL your system. Even if you use the SETSSI command so that changes take effect immediately, you should add the entries to the IEFSSNss member of SYS1.PARMLIB as well, so that they will remain in effect after subsequent IPLs.

There are two methods of doing this:
- The keyword parameter form of the MQSeries subsystem definition in IEFSSNss. This is the recommended method.

```
SUBSYS SUBNAME(ssid) INITRTN(CSQ3INI) INITPARM('CSQ3EPX,cpf,scope')
```

- The positional parameter form of the MQSeries subsystem definition.

```
ssid,CSQ3INI,'CSQ3EPX,cpf,scope'
```

Do not mix the two forms in one IEFSSNss member. If different forms are required, use a separate IEFSSNss member for each type, adding the SSN operand of the new member to the IEASYSxx SYS1.PARMLIB member. To specify more than one SSN, use SSN=(aa,bb,...) in IEASYSxx.

The corresponding SETSSI command to dynamically define an MQSeries subsystem is:

```
SETSSI ADD,S=ssid,I=CSQ3INI,P='CSQ3EPX,cpf,scope'
```

In the examples above,

**ssid** The subsystem identifier.

| **cpf** The command prefix string (see "Defining command prefix strings" on page 13 for information about CPFs).

| **scope** The system scope, used if you are running in an OS/390 sysplex (see "CPFs in a sysplex environment" on page 14 for information about system scope).

Figure 1 shows several examples.

```
CSQ1,CSQ3INI,'CSQ3EPX,+mqs1cpf,S'
CSQ2,CSQ3INI,'CSQ3EPX,+mqs2cpf,S'
CSQ3,CSQ3INI,'CSQ3EPX,++,S'
```

*Figure 1. Sample IEFSSNss statements for defining subsystems*

**Note:** Once you have created objects in a subsystem, you cannot change the subsystem name or use the page sets from one subsystem in another subsystem. To do either of these, you must unload all the objects and messages from one subsystem and reload them into another.

Table 4 gives a number of examples showing the associations of subsystem names and CPFs, as defined by the statements in Figure 1.

*Table 4. Subsystem name to CPF associations*

| **MQSeries subsystem name** | **CPF** |
|---|---|
| CSQ1 | +mqs1cpf |
| CSQ2 | +mqs2cpf |
| CSQ3 | ++ |

**Note:** The ACTIVATE and DEACTIVATE functions of the SETSSI OS/390 command are not supported by MQSeries.

## Defining command prefix strings

You should adopt a system-wide convention for your CPFs for all subsystems to avoid conflicts. You should adhere to the following guidelines:

- Define a CPF as a one- to eight-character string.
- Do not use a CPF that is already in use by any other subsystem, and avoid using the JES backspace character defined on your system as the first character of your string.
- Define your CPF using characters from the set of valid characters listed in Table 6 on page 14.
- Do not use a CPF that is an abbreviation for an already defined process or that might be confused with command syntax. For example, a CPF such as ‘D’ conflicts with OS/390 commands such as DISPLAY. To avoid this happening, you should use one of the special characters (shown in Table 6 on page 14) as the first or only character in your CPF string.
- Do not define a CPF that is either a subset or a superset of an existing CPF. For an example, see Table 5:

*Table 5. Example of CPF subset and superset rules*

| **Subsystem name** | **CPF defined** | **Commands routed to...** |
|---|---|---|
| MQA | !A | MQA |
| MQB | !B | MQB |
| MQC1 | !C1 | MQC1 |
| MQC2 | !C2 | MQC2 |
| MQB1 | !B1 | **MQB** |

## Define the subsystem

Commands intended for subsystem MQB1 (using CPF !B1) are routed to subsystem MQB because the CPF for this subsystem is !B, a subset of !B1. For example, if you entered the command `!B1 START QMGR`, subsystem MQB will receive the command `1 START QMGR` (which, in this case, it will be unable to deal with).

You can see which prefixes already exist by issuing the OS/390 command DISPLAY OPDATA.

If you are running in a sysplex, OS/390 will diagnose any conflicts of this type at the time of CPF registration (see “CPFs in a sysplex environment” for information about CPF registration).

Table 6 shows the characters that you can use when defining your command prefix (CPF) strings:

*Table 6. Valid character set for CPF strings*

| **Character set** | **Contents** |
|---|---|
| Alphabetic | Uppercase A through Z, lowercase a through z |
| Numeric | 0 through 9 |
| National (see note) | @ $ # (Characters that can be represented as hexadecimal values X'7C', X'5B', and X'7B') |
| Special | . / ( ) * & + - = ¢ < \| ! ; % _ ? : |
| **Note:** The system recognizes the following hexadecimal representations of the national characters: @ as X'7C', $ as X'5B', and # as X'7B'. In countries other than the U.S., the U.S. national characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error. For example, in some countries the $ character might generate an X'4A'. | |

### CPFs in a sysplex environment

If you are in a sysplex environment, MQSeries registers your CPFs to enable you to enter a command from any console in the sysplex and route that command to the appropriate system for execution. The command responses are returned to the originating console.

**Defining the scope for sysplex operation:** Scope is used to determine the type of CPF registration performed by the MQSeries subsystem when you are running MQSeries in a sysplex environment.

Possible values for scope are as follows:

**M** System scope.

The CPF is registered with OS/390 at system IPL time by MQSeries and remains registered for the entire time that the OS/390 system is active.

MQSeries operator commands must be entered at a console connected to the OS/390 image running the target subsystem, or you must use ROUTE commands to direct the command to that image.

You should use this option if you are not running in a sysplex.

**X** Sysplex IPL scope.

The CPF is registered with OS/390 at system IPL time by MQSeries and remains registered for the entire time that the OS/390 system is active.

MQSeries operator commands can be entered at any console connected to the sysplex, and are routed to the image that is executing the target system automatically.

**S** Sysplex started scope.

The CPF is registered with OS/390 at the time the MQSeries subsystem is started and remains active until the MQSeries subsystem terminates.

You must use ROUTE commands to direct the original START QMGR command to the target system, but all further MQSeries operator commands can be entered at any console connected to the sysplex, and are routed to the target system automatically.

After MQSeries termination, you must use the ROUTE commands to direct subsequent START commands to the target MQSeries subsystem.

An MQSeries subsystem with a CPF with scope of X can only be defined on one OS/390 image within a sysplex. If you use this option, you must define a unique subsystem name table for each OS/390 image requiring MQSeries subsystems with CPFs of scope X.

An MQSeries subsystem with a CPF with scope of S can be defined on one or more OS/390 images within a sysplex, so these images can share a single subsystem name table. However you must ensure that the initial START command is issued on (or routed to) the OS/390 image on which you want the MQSeries subsystem to run. If you use this option, you can stop the MQSeries subsystem and restart it on a different OS/390 image within the sysplex without having to change the subsystem name table or re-IPL an OS/390 system.

If you want to use the OS/390 automatic restart manager (ARM) to restart queue managers in different OS/390 images automatically, every queue manager must be defined in each OS/390 image on which that queue manager might be restarted. Every queue manager must be defined with a sysplex-wide, unique 4-character subsystem name with a CPF scope of S.

## Task 4: Update the OS/390 link list and LPA

- *You need only perform this task once.*
- *You must IPL the system before these changes take effect.*
- *You might need to perform this task when migrating from a previous version.*

You must add the MQSeries early code library, thlqual.SCSQLINK, to the link list, SYS1.PARMLIB(LNKLSTkk), and put thlqual.SCSQLINK in the master catalog.

If you want to minimize the number of libraries in the link list, copy the load modules from thlqual.SCSQLINK into an existing library that is in the link list and in the master catalog. The library you copy the members into must also be APF-authorized. However, if you do this, the installation program (SMP/E) cannot apply service to these modules, so you must recopy the load modules if service is to be applied to them.

You also need the associated early error message module, CSQ3ECMX. Either add the library containing the language you want to the link list, or copy this module from that library to an existing library in the link list. The libraries are called thlqual.SCSQSNLx, where x is the language letter.

| A user modification that moves the contents of the thlqual.SCSQSNLx data set into
| thlqual.SCSQLINK and informs SMP/E is supplied with MQSeries. It is called
| CSQ8ERLY and is described in the *MQSeries for OS/390 Program Directory*.

The distributed queuing facility, CICS bridge, and Internet Gateway need access to the LE runtime library SCEERUN. You might want to include it in the link list. (This is not required for the CICS mover.)

| All the MQSeries supplied load modules in the following libraries are reentrant
| and can be placed in the LPA if desired:
| • SCSQAUTH
| • SCSQANLC
| • SCSQANLE
| • SCSQANLK
| • SCSQANLU
| • SCSQMVR1
| • SCSQMVR2

| This is particularly recommended for SCSQAUTH so that you do not have to
| include it in several STEPLIBs.

## Task 5: APF authorize the MQSeries load libraries

- *You need only perform this task once unless you are using queue-sharing groups, in which case you should perform this task for each OS/390 in a sysplex.*
- *You might need to IPL the system before these changes take effect.*
- *You might need to perform this task when migrating from a previous version.*

The MQSeries load libraries thlqual.SCSQAUTH and thlqual.SCSQLINK must be APF-authorized. You must also APF-authorize the libraries for your national language feature (thlqual.SCSQANLx and thlqual.SCSQSNLx) and for the non-CICS mover (thlqual.SCSQMVR1 or thlqual.SCSQMVR2).

All members of the link list are APF-authorized if the SYS1.PARMLIB member IEASYSpp contains the statement:

```
LNKAUTH=LNKLST
```

LNKAUTH=LNKLST is the default if LNKAUTH is not specified.

Because thlqual.SCSQLINK must be included in the link list, if IEASYSpp contains this LNKAUTH statement or if you allow it to default, you do not need to put thlqual.SCSQLINK in the APF list as well.

**Note:** You must APF-authorize all the libraries that you include in the MQSeries STEPLIB. If you put a library that is not APF-authorized in the STEPLIB, the whole library concatenation loses its APF authorization.

The APF lists are in the SYS1.PARMLIB member PROGxx or IEAAPFaa. The lists contain the names of APF authorized OS/390 libraries. The order of the entries in the lists is not significant. See the *MVS Initialization and Tuning Reference* manual for information about APF lists.

If you use PROGxx members with dynamic format, you need only issue the `SET PROG=` OS/390 command for the changes to take effect. Otherwise, if you use static format or IEAAPFaa members, you must IPL your system.

## Task 6: Update the OS/390 program properties table

- *You need only perform this task once unless you are using queue-sharing groups, in which case you should perform this task for each OS/390 in a sysplex.*
- *You do not need to IPL the system before these changes take effect.*
- *You do not need to perform this task when migrating from a previous version.*

You must add the following entry to the program properties table (PPT) which you can find in SYS1.PARMLIB(SCHEDxx).

```
PPT  PGMNAME(CSQYASCP) /* CSQ - THIS IS REQUIRED FOR MQSERIES  */
     CANCEL            /* CAN BE CANCELLED                     */
     KEY(7)            /* STORAGE PROTECTION KEY               */
     SWAP              /* PROGRAM IS SWAPPABLE                 */
     NOPRIV            /* NOT PRIVILEGED                       */
     DSI               /* REQUIRES DATA SET INTEGRITY          */
     PASS              /* NOT ALLOWED TO BYPASS PASS PROT      */
     SYST              /* SYSTEM TASK SO NOT TIMED             */
     AFF(NONE)         /* NO PROCESSOR AFFINITY                */
     NOPREF            /* NO PREFERRED STORAGE FRAMES          */
```

*Figure 2. PPT additional entries needed for the MQSeries queue manager*

You should not set NOSWAP because the MQSeries queue manager controls swapping itself. However, if you have a heavily-loaded MQSeries network and response time is critical, it might be advantageous to make the MQSeries channel initiator non-swappable, by adding the following further PPT entry, at the risk of impacting the performance of the rest of your OS/390 system:

```
PPT  PGMNAME(CSQXJST)  /* CSQ - MAKE MQSERIES MOVER NON-SWAPPABLE */
     CANCEL            /* CAN BE CANCELLED                        */
     KEY(8)            /* STORAGE PROTECTION KEY                  */
     NOSWAP            /* PROGRAM IS NON-SWAPPABLE                */
```

*Figure 3. PPT additional entries needed for the MQSeries channel initiator*

Issue the SET SCH= OS/390 command for these changes to take effect.

## Task 7: Create procedures for the MQSeries subsystem

- *Repeat this task for each MQSeries queue manager.*
- *You might need to perform this task when migrating from a previous version.*

For each MQSeries subsystem defined in the subsystem name table, create a cataloged procedure in a procedure library. The IBM-supplied procedure library is called SYS1.PROCLIB, but your installation might use its own naming convention.

The name of the MQSeries started task procedure is formed by concatenating the subsystem name with the characters MSTR. For example, subsystem CSQ1 has the procedure name CSQ1MSTR. You need one procedure for each of the subsystems you define.

We recommend that a subsystem called CSQ1MSTR is created initially for installation verification and testing purposes.

Copy the sample started task procedure thlqual.SCSQPROC(CSQ4MSTR) to member CSQ1MSTR (or a name of your choice) of your SYS1.PROCLIB or, if you are not using SYS1.PROCLIB, your procedure library. Copy CSQ4MSTR to a member in your procedure library for each of the MQSeries subsystems that you define.

When you have copied the members, you can tailor them to the requirements of each subsystem, using the instructions in member CSQ4MSTR. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started. This is described with the start options in the *MQSeries for OS/390 System Administration Guide*.

You must concatenate thlqual.SCSQANLx (where x is the language letter for your national language) before thlqual.SCSQAUTH in the STEPLIB DD statement.

| If you are using queue-sharing groups, the STEPLIB concatenation must include
| the DB2 runtime target library. In the *DB2 Installation Guide*, this is known as the
| prefix.SDSNLOAD library and it must be APF-authorized. This library is only
| required in the STEPLIB concatenation if it is not accessible through the linklist.

Before you start MQSeries, you should set up MQSeries data set and system security by:

- Authorizing the queue manager started task procedure to run under your external security manager.
- Authorizing access to the queue manager data sets.

For details about how to do this, see “Security installation tasks” on page 195.

The exit library (CSQXLIB) can be added to this procedure later if you want to use queue manager exits. In this case, you also need access to the LE run-time library SCEERUN; if it is not in your link list (SYS1.PARMLIB(LNKLSTkk)), concatenate it in the STEPLIB DD statement. You need to stop and restart your queue manager to do this.

## Task 8: Create procedures for the channel initiator

- *Repeat this task for each MQSeries queue manager.*
- *Omit this task if you are using the CICS mover.*
- *You might need to perform this task when migrating from a previous version.*

You need to create a channel-initiator started task procedure for each MQSeries subsystem that is going to use distributed queuing.

To do this, you need to:

1. Copy the sample started task procedure thlqual.SCSQPROC(CSQ4CHIN) to your procedure library. Name the procedure *xxxx*CHIN, where *xxxx*. is the name of your MQSeries subsystem (for example, CSQ1CHIN would be the channel initiator started task procedure for queue manager CSQ1).
2. Make a copy for each MQSeries subsystem that you are going to use.
3. Tailor the procedures to your requirements using the instructions in the sample procedure CSQ4CHIN. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started. This is described with the start options in the *MQSeries for OS/390 System Administration Guide*.

   Concatenate the library containing your national language feature (thlqual.SCSQANLx where x is the letter for your language) before thlqual.SCSQAUTH in the STEPLIB DD statement.

   Choose the appropriate distributed queuing library: thlqual.SCSQMVR1 if you are using the OpenEdition sockets or IUCV TCP/IP interface, or thlqual.SCSQMVR2 if you are using the SOLVE:TCPaccess interface. For LU 6.2 you can use either library.

   Access to the LE run-time library SCEERUN is required; if it is not in your link list (SYS1.PARMLIB(LNKLSTkk)), concatenate it in the STEPLIB DD statement.
4. Authorize the procedures to run under your external security manager.

The exit library (CSQXLIB) can be added to this procedure later if you want to use channel exits. You will need to stop and restart your channel initiator to do this.

If you are using TCP/IP, the channel initiator address space must be able to access the TCPIP.DATA data set that contains TCP/IP system parameters. There are various ways that this has to be set up, depending on which TCP/IP product and interface you are using. They include:

- Environment variable, RESOLVER_CONFIG
- HFS file, /etc/resolv.conf
- //SYSTCPD DD statement
- //SYSTCPDD DD statement
- *jobname/userid*.TCPIP.DATA
- SYS1.TCPPARMS(TCPDATA)
- *zapname*.TCPIP.DATA

Some of these will affect your started-task procedure JCL. For more information, see the following:

- *TCP/IP OpenEdition: Planning and Release Guide*
- *OS/390 OpenEdition: Planning*
- Your SOLVE:TCPaccess documentation

## Task 9: Set up the DB2 environment

- *Repeat this task for each DB2 data-sharing group.*
- *You need to perform this task when migrating from a previous version.*
- *Omit this task if you are not using queue-sharing groups.*

You need to establish an environment in which MQSeries can access and execute the DB2 plans that are used for queue-sharing groups.

The following steps must be performed for each DB2 data-sharing group. All the sample JCL is in thlqual.SCSQPROC.

1. Customize and execute sample JCL CSQ45CSG to create the storage group that is to be used for the MQSeries database, tablespaces, and tables.
2. Customize and execute sample JCL CSQ45CDB to create the database to be used by all queue managers that will be connecting to this DB2 data-sharing group.
3. Customize and execute sample JCL CSQ45CTS to create the tablespaces that will contain the queue manager and channel initiator tables that are used for queue-sharing groups (to be created in step 4).
4. Customize and execute sample JCL CSQ45CTB to create the 10 DB2 tables and associated indexes. Do not change any of the row names or attributes.
5. Customize and execute sample JCL CSQ45BPL to bind the DB2 plans for the queue manager, utilities, and channel initiator.
6. Customize and execute sample JCL CSQ45GEX to grant execute authority to the respective plans for the user IDs that will be used by the queue manager, utilities, and channel initiator. The user IDs for the queue manager and channel initiator are the user IDs under which their started task procedures run. The user IDs for the utilities are the user IDs under which the batch jobs can be submitted. The names of the appropriate plans are:

| **User** | **Plans** |
|---|---|
| Queue manager | CSQ5D220, CSQ5L220, CSQ5R220, CSQ5U220, CSQ5W220 |
| Channel initiator | CSQ5S220, CSQ5K220 |
| SDEFS function of the CSQUTIL batch utility | CSQ52220 |
| CSQ5PQSG batch utility | CSQ5B220 |

In the event of a failure during DB2 setup, the following jobs can be customized and executed:

- CSQ45DTB to drop the tables and indexes.
- CSQ45DTS to drop the tablespaces.
- CSQ45DSG to drop the storage group.
- CSQ45DDB to drop the database.

See the *DB2 for OS/390 Administration Guide* for more information about setting up DB2.

## Task 10: Set up the Coupling Facility

- *Repeat this task for each queue-sharing group.*
- *You need to perform this task when migrating from a previous version.*
- *Omit this task if you are not using queue-sharing groups.*

You need to define the Coupling Facility structures used by the queue managers in the queue-sharing group in the Coupling Facility Resource Management (CFRM) policy data set, using IXCMQAPU.

All the structures for the queue-sharing group start with the name of the queue-sharing group. You must have:

- An administrative structure called *qsg-name*CSQ_ADMIN. This structure is used by MQSeries itself and does not contain any user data.
- One or more structures used to hold messages for shared queues. These can have any name you choose up to 16 characters long.
  - The first four characters must be the queue-sharing group name.
  - The fifth character must be alphabetic and subsequent characters can be alphabetic or numeric. This part of the name (without the queue-sharing group name) is what you specify for the CFSTRUCT attribute when you define a shared queue.

  You can use only alphabetic and numeric characters in the names of structures used to hold messages for shared queues, you cannot use any other characters (for example, the _ character, which is used in the name of the administrative structure).

The structures for shared queue messages must be defined to the Coupling Facility in this way before they can be used. The *MQSeries for OS/390 Concepts and Planning Guide* explains how to decide how large to make your structures.

Sample control statements for IXCMIAPU are in data set thlqual.SCSQPROC(CSQ4CFRM). Customize these and add them to your IXCMIAPU job for the Coupling Facility and run it.

When you have defined your structures successfully, you need to activate the CFRM policy that is being used. To do this, issue the following OS/390 command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=policy-name
```

## Task 11: Implement your ESM security controls

- *Repeat this task for each MQSeries queue manager or queue-sharing group.*
- *You might have to IPL the system before these changes take effect.*
- *You might need to perform this task when migrating from a previous version.*

You must now consider how you are going to implement any security controls for MQSeries.

If you use RACF as your external security manager, see “Part 5. Setting up security” on page 135, which describes how to implement these security controls. If you are using RACF, you might need to IPL the system if you change the started-task procedures table, depending on how you choose to associate user IDs with the started tasks.

| If you are using queue-sharing groups, you should ensure that the user IDs
| associated with the queue manager, channel initiator, and the utilities (as specified
| in step 6 on page 21) have authority to establish a RRSAF connection to each of the
| DB2 subsystems with which you want to establish a connection. The RACF profile
| to which the user ID requires READ access is *DB2ssid*.RRSAF in the DSNR
| resource class.

If you are using the channel initiator, you must also do the following tasks:

1. If your subsystem has connection security active, define a connection security profile ssid.CHIN to your external security manager (see “Connection security profiles for distributed queuing” on page 149 for information about this).
2. If you are using a sockets interface, ensure that the user ID under whose authority the channel initiator is running is configured to use OpenEdition, as described in the *OS/390 OpenEdition Planning* manual.

| Those queue managers that will access the Coupling Facility list structures require
| the appropriate security access. The RACF class is FACILITY. The queue manager
| user ID requires ALTER access to the IXLSTR.*structure-name* profile.

## Task 12: Customize the initialization input data sets

- *Repeat this task for each MQSeries queue manager.*
- *You need to perform this task when migrating from a previous version.*

Each MQSeries instance gets its initial definitions from a series of commands contained in the MQSeries *initialization input data sets*. These data sets are referenced by the DDnames CSQINP1 and CSQINP2 defined in the MQSeries subsystem started task procedure.

Responses to these commands are written to the initialization output data sets referenced by the DDnames CSQOUT1 and CSQOUT2.

| Sample initialization input data sets are supplied with MQSeries, see the *MQSeries*
| *for OS/390 Concepts and Planning Guide* for information about them.

To preserve the originals, you should make working copies of each sample. Then you can tailor the commands in these working copies to suit your system requirements.

If you intend to define more than one MQSeries subsystem, you are recommended to include the subsystem name in the high-level qualifier of the initialization input data set name. This allows you to identify the MQSeries subsystem associated with each data set more easily.

### Initialization data set formats

The initialization input data sets can be partitioned data set (PDS) members or sequential data sets. They can be a concatenated series of data sets. Define them with a record length of 80 bytes, where:

- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk (*) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.
- Each command must start on a new record.
- A trailing − means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The maximum number of characters permitted in a command is 32 762.

If you use a sequential data set for CSQINP1 or CSQINP2, the data set remains allocated to the queue manager started task while the queue manager is active. During this time, it is not available for editing; if you want to change the data set, you must first stop the queue manager. The same applies to CSQINPX for the duration of the channel initiator started task.

The initialization output data sets are sequential data sets, with a record length of 125, a record format of VBA, and a block size of 629.

## Using the CSQINP1 sample

The sample CSQINP1 data set thlqual.SCSQPROC(CSQ4INP1) contains definitions of buffer pools, page set to buffer pool associations, MAXSMSGS, and an ALTER SECURITY command. The sample should be included in the CSQINP1 concatenation of your MQSeries started task procedure.

**Notes:**

1. MQSeries supports up to four buffer pools (0 through 3). The DEFINE BUFFPOOL command can only be issued from a CSQINP1 initialization data set. The definitions in the sample specify four buffer pools.
2. Each page set used by the subsystem must be defined in the CSQINP1 initialization data set by using the DEFINE PSID command. The page set definition associates a buffer pool ID with a page set. If no buffer pool is specified, buffer pool 0 is used by default.

   Page set zero (00) must be defined. It contains all the object definitions. You can define up to 100 page sets for each MQSeries subsystem.
3. The DEFINE MAXSMSGS command defines the maximum number of **MQGET** and **MQPUT** calls that can be made within an MQSeries unit of recovery. In CSQ4INP1, the default value for MAXSMSGS is defined as 10 000.
4. The ALTER SECURITY command can be used to alter the security attributes TIMEOUT and INTERVAL. In CSQ4INP1, the default values are defined as 54 and 12 respectively.

See the *MQSeries for OS/390 Concepts and Planning Guide* for information about organizing buffer pools and page sets.

## Using the CSQINP2 samples

| Samples thlqual.SCSQPROC(CSQ4INSG) and thlqual.SCSQPROC(CSQ4INSX)
| contain system object definitions that can be included in the CSQINP2
| concatenation of your MQSeries started task procedure. If you are using
| queue-sharing groups, you also need to customize and include sample
| thlqual.SCSQPROC(CSQ4INSS).

Samples thlqual.SCSQPROC(CSQ4INYG), thlqual.SCSQPROC(CSQ4INYD), and thlqual.SCSQPROC(CSQ4INYC) contain some definitions that you can customize for your own objects.

| You need to define objects once only, not each time that you start a queue
| manager, so it is not necessary to include these definitions in CSQINP2 every time.
| If you do include them every time, you will attempt to define objects that already
| exist, and you will get messages similar to the following:

```
CSQM095I +CSQ1 CSQMMSGP QLOCAL(SYSTEM.DEFAULT.LOCAL.QUEUE) ALREADY EXISTS
CSQM090E +CSQ1 CSQMMSGP FAILURE REASON CODE X'00D44003'
CSQ9023E +CSQ1 CSQMMSGP ' DEFINE QLOCAL' ABNORMAL COMPLETION
```

The objects are not damaged by this failure. If you want to leave the SYSTEM definitions data set in the CSQINP2 concatenation, you can avoid the failure messages by specifying the REPLACE attribute against each object.

## Using the CSQINPX sample

Sample thlqual.SCSQPROC(CSQ4INPX) contains a set of commands that you might want to execute each time the channel initiator starts. You must customize this sample before use; you can then include it in the CSQINPX data set for the channel initiator.

The MQSC commands contained in the data set are executed at the end of channel initiator initialization, and output is written to the data set specified by the CSQOUTX DD statement. The output is similar to that produced by the COMMAND function of the MQSeries utility program (CSQUTIL). See the *MQSeries for OS/390 System Administration Guide* for information about the MQSeries utility program.

You can specify any of the MQSC commands that can be issued from CSQUTIL, not only the channel commands. You can enter commands from other sources while CSQINPX is being processed. All commands are issued in sequence, regardless of the success of the previous command.

To specify a command response time, you can use the pseudo-command COMMAND as the first command in the data set. This takes a single optional keyword RESPTIME(*nnn*), where *nnn* is the time, in seconds, to wait for a response to each of the commands. This is in the range 5 through 999; the default is 30.

If MQSeries detects that the responses to four commands have taken too long, processing of CSQINPX is stopped and no further commands are issued. The channel initiator is not stopped, but message CSQU052E is written to the CSQOUTX data set, and message CSQU013E is sent to the console.

When MQSeries has completed processing of CSQINPX successfully, message CSQU012I is sent to the console.

## Task 13: Create the bootstrap and log data sets

- *Repeat this task for each MQSeries queue manager*
- *You do not need to perform this task when migrating from a previous version.*

Use the supplied program CSQJU003 to prepare the bootstrap data sets (BSDSs) and log data sets. You must run this job once for each subsystem you want to define. The sample JCL and Access Method Services (AMS) control statements to run CSQJU003 to create a single or dual logging environment are held in thlqual.SCSQPROC(CSQ4BSDS). Customize and run this job to create your BSDSs and logs.

The startup procedure, CSQ4MSTR, described in "Task 7: Create procedures for the MQSeries subsystem" on page 19, refers to BSDSs in statements of the form:

```
//BSDS1     DD DSN=++HLQ++.BSDS01,DISP=SHR
//BSDS2     DD DSN=++HLQ++.BSDS02,DISP=SHR
```

The log data sets are referred to by the BSDSs.

**Notes:**

1. The BLKSIZE must be specified on the SYSPRINT DD statement in the CSQTLOG step. The BLKSIZE must be 629.
2. To help identify bootstrap data sets and log data sets from different MQSeries subsystems, include the subsystem name in the high level qualifier of these data sets.

| See the *MQSeries for OS/390 Concepts and Planning Guide* for information about
| bootstrap and log data sets.

## Task 14: Define your page sets

- *Repeat this task for each MQSeries queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

| You must define separate page sets for each MQSeries subsystem.
| thlqual.SCSQPROC(CSQ4PAGE) contains JCL and AMS control statements to
| define and format page sets. The JCL runs the supplied utility program CSQUTIL.
| Review the samples and customize them for the number of page sets you want
| and the sizes to use. See the *MQSeries for OS/390 Concepts and Planning Guide* for
| information about page sets and how to calculate suitable sizes.

The startup procedure CSQ4MSTR described in “Task 7: Create procedures for the MQSeries subsystem” on page 19 refers to the page sets, in a statement of the form:

```
//CSQP00nn  DD DISP=OLD,DSN=xxxxxxxxx
```

where *nn* is the page set number between 00 and 99, and *xxxxxxxxx* is the data set that you define.

**Notes:**

1. If you intend to use the dynamic page set expansion feature, ensure that secondary extents are defined for each page set. thlqual.SCSQPROC(CSQ4PAGE) shows how to do this.
2. To help identify page sets from different MQSeries subsystems, include the subsystem name in the high level qualifier of the data set associated with each page set.
3. If you intend to allow the FORCE option to be used with the FORMAT function of the utility program CSQUTIL, you must add the REUSE attribute on the AMS DEFINE CLUSTER statement. This is described in the *MQSeries for OS/390 System Administration Guide*.

## Task 15: Add the MQSeries entries to the DB2 data-sharing group

- *Repeat this task for each MQSeries queue-sharing group and each queue manager.*
- *You need to perform this task when migrating from a previous version.*
- *Omit this task if you are not using queue-sharing groups.*

If you are using queue-sharing groups, run the CSQ5PQSG utility to add queue-sharing group and queue manager entries to the MQSeries tables in the DB2 data-sharing group. Run the utility for each queue-sharing group and each queue manager that is to be a member of a queue-sharing group. (CSQ5PQSG is described in the *MQSeries for OS/390 System Administration Guide*.)

You should perform the following actions in the specified order:

1. Add a queue-sharing group entry into the MQSeries DB2 tables using the ADD QSG function of the CSQ5PQSG program. A sample is provided in thlqual.SCSQPROC(CSQ45AQS).

   Perform this function once for each queue-sharing group that is defined in the DB2 data-sharing group. The queue-sharing group entry must exist before adding any queue manager entries that reference the queue-sharing group.
2. Add a queue manager entry into the MQSeries DB2 tables using the ADD QMGR function of the CSQ5PQSG program. A sample is provided in thlqual.SCSQPROC(CSQ45AQM).

   Perform this function for each queue manager that is to be a member of the queue-sharing group.

   **Note:** A queue manager can only be a member of one queue-sharing group.

## Task 16: Tailor your system parameter module

- *Repeat this task for each MQSeries queue manager, as required.*
- *You need to perform this task when migrating from a previous version.*

The MQSeries system parameter module controls the logging, archiving, tracing, and connection environments that MQSeries uses in its operation. The system parameter module has three macros as follows:

| **Macro name** | **Purpose** |
|---|---|
| CSQ6SYSP | Specifies the connection and tracing parameters, see page 31 |
| CSQ6LOGP | Controls log initialization, see page 38 |
| CSQ6ARVP | Controls archive initialization, see page 42 |

MQSeries supplies a default system parameter module, CSQZPARM, which is invoked automatically if you issue the START QMGR command (without a PARM parameter) to start an instance of MQSeries. CSQZPARM is in the APF-authorized library thlqual.SCSQAUTH also supplied with MQSeries. The values of these parameters are displayed as a series of messages when you start MQSeries.

See the *MQSeries MQSC Command Reference* manual for more information about the START QMGR command and the *MQSeries for OS/390 System Administration Guide* for more information about how this command is used.

### Creating your own system parameter module

If CSQZPARM does not contain the system parameters you want, you can create your own system parameter module using the sample JCL provided in thlqual.SCSQPROC(CSQ4ZPRM).

To create your own system parameter module:

1. Make a working copy of the JCL sample.
2. Edit the parameters for each macro in the copy as required. If you remove any parameters from the macro calls, the default values are automatically picked up at run time.
3. Replace the placeholder `++NAME++` with the name that the load module is to take (this can be CSQZPARM).
4. If your assembler is not high level assembler, change the JCL as required by your assembler.
5. Run the JCL to assemble and link-edit the tailored versions of the system parameter macros to produce a load module. This is the new system parameter module with the name that you have specified.
6. Put the load module produced in an APF-authorized user library.
7. Include this library in the MQSeries started task procedure STEPLIB. This library name must come before the library thlqual.SCSQAUTH in STEPLIB.
8. Invoke the new system parameter module when you start MQSeries. For example, if the new module is named NEWMODS, issue the command:
   ```
   START QMGR PARM(NEWMODS)
   ```

**Note:** If you choose to name your module CSQZPARM, you do not need to specify the PARM parameter on the START QMGR command.

## Fine tuning a system parameter module

MQSeries also supplies a set of three assembler source modules, which can be used to fine tune an existing system parameter module. These modules are in library thlqual.SCSQASMS. Typically, you use these modules in a test environment to change the default parameters in the system parameter macros. Each source module calls a different system parameter macro:

| This assembler source module... | Calls this macro... |
|---|---|
| CSQFSYSP | CSQ6SYSP (connection and tracing parameters) |
| CSQJLOGP | CSQ6LOGP (log initialization) |
| CSQJARVP | CSQ6ARVP (archive initialization) |

This is how you use these modules:

1. Make working copies of each assembler source module in a user assembler library.
2. Edit your copies by adding or altering the values of any parameters as required.
3. Assemble your copies of any edited modules to create object modules in a user object library.
4. Link-edit these object code modules with an existing system parameter module to produce a load module that is the new system parameter module.
5. Ensure that new system parameter module is a member of a user authorized library.
6. Include this library in the MQSeries started task procedure STEPLIB. This library must come before the library thlqual.SCSQAUTH in STEPLIB.
7. Invoke the new system parameter module by issuing a START QMGR command, specifying the new module name in the PARM parameter, as before.

## Using CSQ6SYSP

Use CSQ6SYSP to set system parameters.

The default parameters for CSQ6SYSP are shown in Table 7. If you want to change any of these values, refer to the detailed descriptions of the parameters.

*Table 7. Default values of CSQ6SYSP parameters*

| Parameter | Description | Default value |
|---|---|---|
| CMDUSER | The default user ID for command security checks. | CSQOPR |
| CTHREAD | Maximum number of connections from batch, CICS, IMS, and TSO tasks to a single instance of MQSeries. | 300 |
| EXITLIM | Time (in seconds) for which queue-manager exits can execute during each invocation. | 30 |
| EXITTCB | How many started server tasks to use to run queue manager exits. | 8 |
| IDBACK | Maximum number of connections to a single instance of MQSeries from batch or TSO background tasks. | 20 |

Table 7. Default values of CSQ6SYSP parameters (continued)

| Parameter | Description | Default value |
|---|---|---|
| IDFORE | Maximum number of connections to a single instance of MQSeries from TSO foreground tasks. | 100 |
| LOGLOAD | Number of log records written by MQSeries between the start of one checkpoint and the next. | 500 000 |
| OTMACON | OTMA connection parameters. | See below |
| QMCCSID | Coded character set identifier for the queue manager. | 0 |
| QSGDATA | Queue-sharing group parameters. | See below |
| RESAUDIT | RESLEVEL auditing parameter. | YES |
| ROUTCDE | Message routing code assigned to messages not solicited from a specific console. | 1 |
| SERVICE | Reserved for use by IBM. | 0 |
| SMFACCT | Specifies whether SMF accounting data is to be collected when MQSeries is started. | NO |
| SMFSTAT | Specifies whether SMF statistics are to be collected when MQSeries is started. | NO |
| STATIME | Default time, in minutes, between each gathering of statistics. | 30 |
| TRACSTR | Specifies whether tracing is to be started automatically. | NO |
| TRACTBL | Size of trace table, in 4 KB blocks, to be used by the global trace facility. | 99 (396 KB) |
| WLMTIME | Time (in minutes) between scanning the queue index for WLM-managed queues. | 30 |

**CMDUSER**
Specifies the default user ID used for command security checks. This user ID must be defined to the ESM (for example, RACF). Specify a name of 1 through 8 alphanumeric characters. The first character must be alphabetic.

The default is CSQOPR.

**CTHREAD**
Specifies the maximum number of connections from batch, CICS, IMS, TSO, and the channel initiator to a single instance of MQSeries.

CICS connections are not limited in the same way as TSO and batch connections. During the connection of the main CICS TCB to MQSeries, the adapter attempts to attach up to eight OS/390 subtasks (TCBs) to be used by this CICS system. This means each CICS system connected takes up nine of the connections specified on CTHREAD, so you must increase CTHREAD by nine for each CICS system connected. You must also increase the value of CTHREAD by one for each instance of the task initiator CKTI.

For IMS connections, the number of connections required is one for the control region, and one for each dependent 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.

For distributed queuing (without CICS), the number of connections required by the channel initiator address space depends on the number of adapter subtasks and dispatchers there will be; see “Using CSQ6CHIP” on page 51.

When the number of connections reaches the limit set by CTHREAD, any further requests for a connection are suspended until a spare slot becomes available. For example, an **MQDISC** call releases that connection. For planning purposes, the value of CTHREAD must be greater than the maximum of IDBACK, IDFORE, and the number of potential connections from the CICS, IMS, and channel initiator address spaces.

Specify a number in the range 1 through 32 767.

The default is 300.

Note that this controls the number of *connections*; a connection might involve more than one thread. Connections and threads are discussed in the *MQSeries for OS/390 Concepts and Planning Guide*.

**EXITLIM**
Specifies the time, in seconds, allowed for each invocation of the queue manager exits. (This parameter has no effect on channel exits.)

Specify a value in the range 5 through 9999.

The default is 30. The queue manager polls exits that are running every 30 seconds. On each poll, any that have been running for more than the time specified by EXITLIM are forcibly terminated.

**EXITTCB**
Specifies the number of started server tasks to use to run exits in the queue manager. (This parameter has no effect on channel exits.)

Specify a value in the range 0 through 99. A value of 0 means that no exits can be run.

The default is 8.

**IDBACK**
Specifies the maximum number of background batch and TSO connections to a single instance of MQSeries. The value of IDBACK is related to those of IDFORE and CTHREAD. See the description of the CTHREAD parameter for more information.

Specify a number in the range 1 through 32 767.

The default is 20.

**IDFORE**
Specifies the maximum number of TSO foreground connections to MQSeries.

The value of IDFORE is related to those of IDBACK and CTHREAD. See the description of the CTHREAD parameter for more information.

The number of TSO connections might be greater than the number of concurrent TSO users if, for example, users split their ISPF screens.

Specify a number in the range 0 through 32 767.

The default is 100.

**LOGLOAD**

Specifies the number of log records that MQSeries writes between the start of one checkpoint and the next. MQSeries starts a new checkpoint after the number of records that you specify has been written.

Specify a value in the range 200 through 16 000 000.

| The default is 500 000.

The greater the value, the better the performance of MQSeries; however, restart takes longer if the parameter is set to a large value.

Suggested settings:

| | |
|---|---|
| **Test system** | 10 000 |
| **Production system** | 500 000<br><br>In a production system, the supplied default value might result in a checkpoint frequency that is too high. |

The value of LOGLOAD determines the frequency of queue manager checkpoints. Too large a value means that a large amount of data is written to the log between checkpoints, resulting in an increased queue manager forward recovery restart time following a failure. Too small a value causes checkpoints to occur too frequently during peak load, adversely affecting response times and CPU usage.

An initial value of 500 000 is suggested for LOGLOAD. (For example, this adds approximately 90 seconds to the restart time after a failure if using RAMAC®® Virtual Array 2 Turbo 82 (RVA2-T82) DASD.) For a 1 KB persistent message rate of 100 messages per second (that is, 100 **MQPUT**s with commit and 100 **MQGET**s with commit) the interval between checkpoints will be approximately 5 minutes.

**Note:** This is intended as a guideline only and the optimum value for this parameter is dependent on the characteristics of the individual system.

**OTMACON**

OTMA parameters. This keyword takes five positional parameters, as shown below:

**OTMACON = (Group,Member,Druexit,Age,TpipePrefix)**

**Group**

This is the name of the XCF group to which this particular instance of MQSeries belongs.

It can be 1 through 8 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that MQSeries should not attempt to join an XCF group.

**Member**
This is the member name of this particular instance of MQSeries within the XCF group.

It can be 1 through 16 characters long and must be entered in uppercase characters.

The default is the 4-character queue manager name.

**Druexit**
This specifies the name of the OTMA destination resolution user exit to be run by IMS.

It can be 1 through 8 characters long.

The default is DFSYDRU0.

This parameter is optional; it is required if MQSeries is to receive messages from an IMS application that was not started by MQSeries. The name should correspond to the destination resolution user exit coded in the IMS system. For more information see “Appendix B. Using OTMA exits in IMS” on page 237.

**Age** This represents the length of time, in seconds, that a user ID from MQSeries is considered previously verified by IMS.

It can be in the range 0 through 2 147 483 647.

The default is 2 147 483 647.

You are recommended to set this parameter in conjunction with the `interval` parameter of the ALTER SECURITY command in order to maintain consistency of security cache settings across the mainframe.

**TpipePrefix**
This represents the prefix to be used for Tpipe names.

It comprises three characters; the first character is in the range A through Z, subsequent characters are A through Z or 0 through 9. The default is CSQ.

This is used each time MQSeries creates a Tpipe; the rest of the name is assigned by MQSeries. You cannot set the full Tpipe name for any Tpipe created by MQSeries.

**QMCCSID**
Specifies the default coded character set identifier that the queue manager (and therefore distributed queuing) is to use.

Specify a value in the range 0 through 65 535. 0 means use the CCSID currently set or, if none is set, use CCSID 500.

The default is 0.

**QSGDATA**

Queue-sharing group data. This keyword takes four positional parameters, as shown below:

**QSGDATA=(`Qsgname,Dsgname,Db2name,Db2servers`)**

**Qsgname**

This is the name of the queue-sharing group to which the queue manager belongs.

It can be 1 through 4 characters long. Acceptable characters are uppercase A-Z, 0-9, $, %, and @. If you specify fewer than 4 characters, each blank is replaced with an '@' symbol.

The default is blanks, which indicates that the queue manager is not a member of any queue-sharing group.

**Dsgname**

This is the name of the DB2 data-sharing group to which the queue manager is to connect.

It can be 1 through 8 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that you are not using queue-sharing groups.

**Db2name**

This is the name of the DB2 subsystem or group attachment to which the queue manager is to connect.

It can be 1 through 4 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that you are not using queue-sharing groups.

**Note:** The DB2 subsystem (or group attachment) must be in the DB2 data-sharing group specified in the `Dsgname`, and all queue managers must specify the same DB2 data-sharing group.

**Db2servers**

This is the number of server tasks used for accessing DB2.

It can be in the range 4 through 10.

The default is 4.

If you specify one of the parameters (apart from the Db2servers parameter), you must enter values for the others, otherwise MQSeries will fail.

**RESAUDIT**

Specifies whether RACF audit records are written for RESLEVEL security checks performed during connection processing.

Specify one of:

| | |
|---|---|
| **NO** | RESLEVEL auditing will not be performed. |
| **YES** | RESLEVEL auditing will be performed. |

The default is YES.

**ROUTCDE**

Specifies the default OS/390 message routing code assigned to messages that are not sent in direct response to an MQSeries command.

Specify one of:

1. A value in the range 1 through 16, inclusive.
2. A list of values, separated by a comma and enclosed in parentheses. Each value must be in the range 1 through 16, inclusive.

The default is 1.

For more information about OS/390 routing codes, see the *MVS Routing and Descriptor Codes* manual.

| **SERVICE**

| This field is reserved for use by IBM.

**SMFACCT**

Specifies whether MQSeries sends accounting data to SMF automatically when MQSeries starts.

Specify one of:

| | |
|---|---|
| **NO** | Do not start gathering accounting data automatically. |
| **YES** | Start gathering accounting data automatically for the default class 1. |

The default is NO.

**SMFSTAT**

Specifies whether to gather SMF statistics automatically when MQSeries starts.

Specify one of:

| | |
|---|---|
| **NO** | Do not start gathering statistics automatically. |
| **YES** | Start gathering statistics automatically for the default class 1. |

The default is NO.

**STATIME**

Specifies the default time, in minutes, between consecutive gatherings of statistics.

| Specify a number in the range 0 through 1440.

| If you specify a value of zero, both statistics data and accounting data will be
| collected at the SMF data collection broadcast. See “Using System Management
| Facility” on page 110 for information about setting this.

The default is 30.

**TRACSTR**
Specifies whether global tracing is to start automatically.

Specify one of:

| | |
|---|---|
| **NO** | Do not start global tracing automatically. |
| **YES** | Start global tracing automatically for the default class, class 1. |
| **integers** | A list of classes for which global tracing is to be started automatically in the range 1 through 4. |
| ***** | Start global trace automatically for all classes. |

The default is NO if you do not specify the keyword in the macro.

**Note:** The supplied default system parameter load module (CSQZPARM) has TRACSTR=YES (set in the assembler module CSQFSYSP). If you do not want to start tracing automatically, you can change this after you have successfully started your MQSeries subsystem. Refer to "Fine tuning a system parameter module" on page 31 for information about how do this.

For details about the START TRACE command, see the *MQSeries MQSC Command Reference* manual.

**TRACTBL**
Specifies the default size, in 4 KB blocks, of trace table where the global trace facility stores MQSeries trace records.

Specify a value in the range 1 through 999.

The default is 99. This is equivalent to 396 KB.

**Note:** Storage for the trace table is allocated in the ECSA. Therefore, you must select this value with care.

**WLMTIME**
Specifies the time (in minutes) between each scan of the indexes for WLM-managed queues.

Specify a value in the range 1 through 9999.

The default is 30.

## Using CSQ6LOGP

Use CSQ6LOGP to establish your logging options.

The default parameters for CSQ6LOGP are shown in Table 8. If you need to change any of these values, refer to the detailed descriptions of the parameters.

*Table 8. Default values of CSQ6LOGP parameters*

| Parameter | Description | Default value |
|---|---|---|
| DEALLCT | Length of time an archive tape unit remains unused before it is deallocated. | 0 |
| INBUFF | Size of input buffer storage for active and archive log data sets. | 60 KB |

*Table 8. Default values of CSQ6LOGP parameters (continued)*

| Parameter | Description | Default value |
|---|---|---|
| MAXARCH | Maximum number of archive log volumes that can be recorded. | 500 |
| MAXRTU | Maximum number of dedicated tape units allocated to read archive log tape volumes concurrently. | 2 |
| OFFLOAD | Archiving on or off. | YES (ON) |
| OUTBUFF | Size of output buffer storage for active and archive log data sets. | 4 000 KB |
| TWOACTV | Single or dual active logging. | YES (dual) |
| TWOARCH | Single or dual archive logging. | YES (dual) |
| TWOBSDS | Single or dual BSDS. | YES (dual BSDS) |
| WRTHRSH | Number of output buffers to be filled before they are written to the active log data sets. | 20 |

**DEALLCT**

Specifies the length of time, in minutes, that an archive read tape unit is allowed to remain unused before it is deallocated.

Specify one of the following:
- Time, in minutes, in the range 0 through 1440
- NOLIMIT

Specifying 1440 or NOLIMIT means that the tape unit will never be deallocated.

The default is 0.

When archive log data is being read from tape, it is recommended that you set this value high enough to allow MQSeries to optimize tape handling for multiple read applications.

When all tape reading is complete, you can alter this value with the SET LOG command. You can display the value of this parameter with the DISPLAY LOG command. These commands are described in the *MQSeries MQSC Command Reference*.

**INBUFF**

Specifies the size, in kilobytes, of the input buffer for reading the active and archive logs during recovery. Use a decimal number in the range 28 through 60. The value specified is rounded up to a multiple of 4.

The default is 60 KB.

Suggested settings:

| | |
|---|---|
| **Test system** | 28 KB |
| **Production system** | 60 KB<br><br>Set this to the maximum for best log read performance. |

**MAXARCH**

Specifies the maximum number of archive log volumes that can be recorded in the BSDS. When this number is exceeded, recording begins again at the start of the BSDS.

Use a decimal number in the range 10 through 1000.

The default is 500.

Suggested settings:

| | |
|---|---|
| **Test system** | 500 (default) |
| **Production system** | 1 000<br><br>Set this to the maximum so that the BSDS can record as many logs as possible. |

For information about the logs and BSDS, see the *MQSeries for OS/390 Concepts and Planning Guide*.

| **MAXRTU**

| Specifies the maximum number of dedicated tape units that can be allocated to
| read archive log tape volumes concurrently.

| This parameter and the DEALLCT parameter allow MQSeries to optimize
| archive log reading from tape devices.

| Specify a value in the range 1 through 99.

| The default is 2.

| It is recommended that you set the value to be at least one less than the
| number of tape units available to MQSeries. If you do otherwise, the offload
| process could be delayed, which could affect the performance of your system.
| For maximum throughput during archive log processing, specify the largest
| value possible for this option, remembering that you need at least one tape
| unit for offload processing.

| You can override this parameter by using the SET LOG command. You can
| display the value of this parameter with the DISPLAY LOG command. These
| commands are described in the *MQSeries MQSC Command Reference*

**OFFLOAD**

Specifies whether archiving is on or off.

Specify either:

| | |
|---|---|
| **YES** | Archiving is on |
| **NO** | Archiving is off |

The parameter cannot be blank.

The default is YES.

**Attention:** Do **not** switch archiving off unless you are working in a test environment. If you do switch it off, you cannot guarantee that data will be recovered in the event of a system or transaction failure.

**OUTBUFF**

Specifies the total size, in kilobytes, of the storage to be used by MQSeries for output buffers for writing the active and archive log data sets. Each output buffer is 4 KB.

The parameter cannot be blank, and must be in the range 40 through 4000. The value specified is rounded up to a multiple of 4.

| The default is 4 000 KB.

Suggested settings:

| | |
|---|---|
| **Test system** | 400 KB |
| **Production system** | 4 000 KB<br>Set this value to the maximum to avoid running out of log output buffers. |

**TWOACTV**

Specifies single or dual active logging.

The parameter cannot be blank. Specify either:

| | |
|---|---|
| **NO** | Single active logs |
| **YES** | Dual active logs |

The default is YES.

| For more information about the use of single and dual logging, refer to the
| *MQSeries for OS/390 Concepts and Planning Guide*.

**TWOARCH**

Specifies the number of archive logs that MQSeries produces when the active log is offloaded.

Specify either:

| | |
|---|---|
| **NO** | Single archive logs |
| **YES** | Dual archive logs |

This parameter cannot be blank even if the OFFLOAD parameter is specified as NO.

The default is YES.

Suggested settings:

| | |
|---|---|
| **Test system** | NO |
| **Production system** | YES (default) |

| For more information about the use of single and dual logging, refer to the
| *MQSeries for OS/390 Concepts and Planning Guide*.

**TWOBSDS**
Specifies the number of bootstrap data sets.

Specify either:

| | |
|---|---|
| **NO** | Single BSDS |
| **YES** | Dual BSDS |

This parameter cannot be left blank.

The default is YES.

| For more information about the use of single and dual logging, refer to the
| *MQSeries for OS/390 Concepts and Planning Guide*.

**WRTHRSH**
Specifies the number of 4 KB output buffers to be filled before they are written to the active log data sets.

The larger the number of buffers, the less often the write takes place, and this improves the performance of MQSeries. The buffers might be written before this number is reached if significant events, such as a commit point, occur.

Specify the number of buffers in the range 1 through 256.

The default is 20.

Suggested settings:

| | |
|---|---|
| **Test system** | 15 |
| **Production system** | 15 |

This is the optimum value. It corresponds to the maximum number of buffers written in a single log I/O.

## Using CSQ6ARVP

Use CSQ6ARVP to establish your archiving environment.

The default parameters for CSQ6ARVP are shown in Table 9. If you need to change any of these values, refer to the detailed descriptions of the parameters. Planning your archive storage is discussed in the *MQSeries for OS/390 Concepts and Planning Guide*.

*Table 9. Default values of CSQ6ARVP parameters*

| Parameter | Description | Default value |
|---|---|---|
| ALCUNIT | Units in which primary and secondary space allocations are made. | BLK (blocks) |
| ARCPFX1 | Prefix for first archive log data set name. | CSQARC1 |
| ARCPFX2 | Prefix for second archive log data set name. | CSQARC2 |
| ARCRETN | The retention period of the archive log data set in days. | 9999 |
| ARCWRTC | List of route codes for messages to the operator about archive log data sets. | 1,3,4 |
| ARCWTOR | Whether to send message to operator and wait for reply before trying to mount an archive log data set. | YES |

Table 9. Default values of CSQ6ARVP parameters (continued)

| Parameter | Description | Default value |
|---|---|---|
| BLKSIZE | Block size of archive log data set. | 20 480 |
| CATALOG | Whether archive log data sets are cataloged in the ICF. | NO |
| COMPACT | Whether archive log data sets should be compacted. | NO |
| PRIQTY | Primary space allocation for DASD data sets. | 4 320 |
| PROTECT | Whether archive log data sets are protected by ESM profiles when the data sets are created. | NO |
| QUIESCE | Maximum time, in seconds, allowed for quiesce when ARCHIVE LOG with MODE(QUIESCE) specified. | 5 |
| SECQTY | Secondary space allocation for DASD data sets. See the ALCUNIT parameter for the units to be used. | 540 |
| TSTAMP | Whether the archive data set name should include a time stamp. | NO |
| UNIT | Device type or unit name on which the first copy of archive log data sets is stored. | TAPE |
| UNIT2 | Device type or unit name on which the second copy of archive log data sets is stored. | Blank |

**ALCUNIT**

Specifies the unit in which primary and secondary space allocations are made.

Specify one of:

| | |
|---|---|
| **CYL** | Cylinders |
| **TRK** | Tracks |
| **BLK** | Blocks |

You are recommended to use BLK because it is independent of the device type.

The default is BLK.

If free space on the archive DASD volumes is likely to be fragmented, you are recommended to specify a smaller primary extent and allow expansion into secondary extents. For more information about space allocation for active logs, refer to the *MQSeries for OS/390 Concepts and Planning Guide*.

**ARCPFX1**

Specifies the prefix for the first archive log data set name.

See the TSTAMP parameter for a description of how the data sets will be named and for restrictions on the length of ARCPFX1.

This parameter cannot be left blank.

The default is CSQARC1.

You might need to authorize the MQSeries subsystem to create archive logs with this prefix.

**ARCPFX2**

Specifies the prefix for the second archive log data set name.

See the TSTAMP parameter for a description of how the data sets will be named and for restrictions on the length of ARCPFX2.

This parameter cannot be blank even if the TWOARCH parameter is specified as NO.

The default is CSQARC2.

You might need to authorize the MQSeries subsystem to create archive logs with this prefix.

**ARCRETN**
Specifies the retention period, in days, to be used when the archive log data set is created.

The parameter must be in the range 0 through 9999.

The default is 9999.

Suggested settings:

| | |
|---|---|
| **Test system** | 3<br><br>In a test system, archive logs will probably not be required over long periods. |
| **Production system** | 9 999 (default)<br><br>Set this value high to effectively switch automatic archive log deletion off. |

Discarding archive log data sets is discussed in the *MQSeries for OS/390 System Administration Guide*.

**ARCWRTC**
Specifies the list of OS/390 routing codes for messages about the archive log data sets to the operator. This field is ignored if ARCWTOR is set to NO.

Specify up to 14 routing codes, each with a value in the range 1 through 16. You must specify at least one code. Separate codes in the list by commas, not by blanks.

The default is the list of values: 1,3,4.

For more information about OS/390 routing codes, see the *MVS Routing and Descriptor Codes* manual.

**ARCWTOR**
Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set.

Other MQSeries users might be forced to wait until the data set is mounted, but they are not affected while MQSeries is waiting for the reply to the message.

Specify either:

**YES** The device needs a long time to mount archive log data sets. For example, a tape drive.

**NO** The device does not have long delays. For example, DASD.

The default is YES.

Suggested settings:

| | |
|---|---|
| **Test system** | NO |
| **Production system** | YES (default)<br><br>This is dependent on operational procedures. If tape robots are used, NO might be more appropriate. |

**BLKSIZE**

Specifies the block size of the archive log data set. The block size you specify must be compatible with the device type you specify in the UNIT parameter.

The parameter must be in the range 4 097 through 28 672. The value you specify is rounded up to a multiple of 4 096.

The default is 20 480.

This parameter is ignored for data sets that are managed by the storage management subsystem (SMS).

If the archive log data set is written to DASD, you are recommended to choose the maximum block size that will allow 2 blocks per track. For example, for a 3390 device, you should use a block size of 24 576.

If the archive log data set is written to tape, specifying the largest possible block size improves the speed of reading the archive log.

Suggested settings:

| | |
|---|---|
| **Test system** | 24 576 (6 log records per block)<br><br>This is the optimum block size for 3390 DASD. |
| **Production system** | 28 672 (the maximum allowed; 7 log records per block)<br><br>Use the highest possible block size for optimum tape I/O efficiency. |

**CATALOG**

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (ICF) catalog.

Specify either:

**NO** Archive log data sets are not cataloged
**YES** Archive log data sets are cataloged

This parameter cannot be blank.

The default is NO.

All archive log data sets allocated on DASD must be cataloged. If you archive to DASD with the CATALOG parameter set to NO, message CSQJ072E is displayed each time an archive log data set is allocated, and MQSeries catalogs the data set.

Suggested settings:

| | |
|---|---|
| **Test system** | YES |
| **Production system** | NO (default) |

**COMPACT**

Specifies whether data written to archive logs is to be compacted. This option applies only to a 3480 or 3490 device that has the improved data recording capability (IDRC) feature. When this feature is turned on, hardware in the tape control unit writes data at a much higher density than normal, allowing for more data on each volume. Specify NO if you do not use a 3480 device with the IDRC feature or a 3490 base model, with the exception of the 3490E. Specify YES if you want the data to be compacted.

Specify either:

| | |
|---|---|
| **NO** | Do not compact the data sets |
| **YES** | Compact the data sets |

The default is NO.

Specifying YES adversely affects performance. Also be aware that data compressed to tape can be read only using a device that supports the IDRC feature. This can be a concern if you have to send archive tapes to another site for remote recovery.

Suggested settings:

| | |
|---|---|
| **Test system** | Not applicable |
| **Production system** | NO (default)<br><br>This applies to 3480 and 3490 IDR compression only. Setting this to YES might degrade archive log read performance during recovery and restart; however, it does not affect writing to tape. |

**PRIQTY**

Specifies the primary space allocation for DASD data sets in ALCUNITs.

The value must be greater than zero.

The default is 4 320.

This value must be sufficient for a copy of either the log data set or its corresponding BSDS, whichever is the larger. To determine the necessary value, follow this procedure:

1. Determine the number of active log records actually allocated (c) as explained in “Task 13: Create the bootstrap and log data sets” on page 27.
2. Determine the number of 4096-byte blocks in each archive log block:

   ```
   d = BLKSIZE / 4096
   ```

   where BLKSIZE is the rounded up value.

3. If ALCUNIT=BLK:

   ```
   PRIQTY = INT(c / d) + 1
   ```

   where INT means round down to an integer.

   If ALCUNIT=TRK:

   ```
   PRIQTY = INT(c / (d * INT(e/BLKSIZE))) + 1
   ```

   where e is the number of bytes per track (56664 for a 3390 device) and INT means round down to an integer.

   If ALCUNIT=CYL:

   ```
   PRIQTY = INT(c / (d * INT(e/BLKSIZE) * f)) + 1
   ```

   where f is the number of tracks per cylinder (5 for a 3390 device) and INT means round down to an integer.

For information about how large to make your log and archive data sets, see "Task 13: Create the bootstrap and log data sets" on page 27 and "Task 14: Define your page sets" on page 28.

Suggested settings:

| | |
|---|---|
| **Test system** | 1 680<br>Sufficient to hold the entire active log, that is:<br>`10 080 / 6 = 1 680 blocks` |
| **Production system** | Not applicable when archiving to tape. |

| If free space on the archive DASD volumes is likely to be fragmented, you are
| recommended to specify a smaller primary extent and allow expansion into
| secondary extents. For more information about space allocation for active logs,
| refer to the *MQSeries for OS/390 Concepts and Planning Guide*.

**PROTECT**

Specifies whether archive log data sets are to be protected by discrete ESM (external security manager) profiles when the data sets are created.

Specify either:

**NO** Profiles are not created.

**YES** Discrete data set profiles are created when logs are offloaded. If you specify YES:
- ESM protection must be active for MQSeries.
- The user ID associated with the MQSeries address space must have authority to create these profiles.
- The TAPEVOL class must be active if you are archiving to tape.

Otherwise, offloads will fail.

The default is NO.

**QUIESCE**

Specifies the maximum time in seconds allowed for the quiesce when an `+CSQ1` ARCHIVE LOG command is issued with MODE QUIESCE specified.

The parameter must be in the range 1 through 999.

The default is 5.

**SECQTY**

Specifies the secondary space allocation for DASD data sets in ALCUNITs.

The parameter must be greater than 0.

The default is 540.

**TSTAMP**

Specifies whether the archive log data set name has a time stamp in it.

Specify either:

**NO** Names do not include a time stamp. The archive log data sets will be named:

*arcpfxi*.A*nnnnnnn*

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 35 characters.

**YES** Names include a time stamp. The archive log data sets will be named:

*arcpfxi*.*cyyddd*.T*hhmmsst*.A*nnnnnnn*

where *c* is 'D' for the years up to and including 1999 or 'E' for the year 2000 and later, and *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 19 characters.

**EXT** Names include a time stamp. The archive log data sets will be named:

*arcpfxi*.D*yyyyddd*.T*hhmmsst*.A*nnnnnnn*

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 17 characters.

The default is NO.

**UNIT**

| Specifies the device type or unit name of the device that is used to store the
| first copy of the archive log data set.

Specify a device type or unit name of 1 through 8 alphanumeric characters. The first character must be alphabetic.

This parameter cannot be blank.

The default is TAPE.

If you archive to DASD, you can specify a generic device type with a limited volume range.

If you archive to DASD:

- Make sure that the primary space allocation is large enough to contain all the data from the active log data sets.
- Make sure that the archive log data set catalog option (CATALOG) is set to YES.
- The archive log data sets cannot extend to another volume.

If you archive to TAPE, MQSeries can extend to a maximum of 20 volumes.

Suggested settings:

| | |
|---|---|
| **Test system** | DASD |
| **Production system** | TAPE |

For more information about choosing a location for archive logs, refer to the *MQSeries for OS/390 Concepts and Planning Guide*.

**UNIT2**

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data sets.

Specify a device type or unit name of 1 through 8 alphanumeric characters. The first character must be alphabetic. If this parameter is blank, the value set for the UNIT parameter is used.

The default is blank.

## Task 17: Tailor the channel initiator parameter module

- *Repeat this task for each MQSeries queue manager, as required.*
- *You need to perform this task when migrating from a previous version.*
- *Omit this task if you are using the CICS mover.*

This process is analogous to tailoring the system parameter module (see “Task 16: Tailor your system parameter module” on page 30).

The channel initiator parameter module controls how distributed queuing operates. It has the single macro, CSQ6CHIP.

MQSeries supplies a default parameter module, CSQXPARM, which is invoked automatically if you issue the START CHINIT command (without a PARM parameter) to start a channel initiator. “Using CSQ6CHIP” on page 51 lists the default values for the supplied CSQXPARM. CSQXPARM is in the APF-authorized library thlqual.SCSQAUTH, also supplied with MQSeries.

The values of these parameters are displayed as a series of messages each time you start the channel initiator.

### Creating your own channel initiator parameter module

In most cases you will need to create your own parameter module. If you are using LU 6.2 communications, you will have to do this because you will at least need to set the outbound LU name to be used. If you are using TCP/IP, you will probably need to set TCPTYPE and TCPNAME.

To create your own parameter module, use the sample JCL provided in thlqual.SCSQPROC(CSQ4XPRM):

1. Make a working copy of the JCL sample.
2. Edit the parameters in the copy as required. See “Using CSQ6CHIP” on page 51 for more information about each parameter. If you remove any parameters from the macro call, the default values are automatically picked up at run time.
3. Replace the placeholder `++NAME++` with the name that the load module is to take. (This can be CSQXPARM.)
4. If your assembler is not high-level assembler, change the JCL as required by your assembler.
5. Run the JCL to assemble and link-edit the tailored versions of the channel initiator parameter macros to produce a load module. This is the new channel initiator parameter module with the name that you have specified.
6. Put the load module produced in an APF-authorized user library.
7. Include this library in the channel initiator started task procedure STEPLIB. This library name must come before the library thlqual.SCSQAUTH in STEPLIB.

8. Invoke the new channel initiator parameter module when you start the channel initiator. For example, if the new module is named NEWMODS, issue the command:

```
START CHINIT PARM(NEWMODS)
```

**Note:** If you choose to name your module CSQXPARM, you do not need to specify the PARM parameter on the START CHINIT command.

## Using CSQ6CHIP

Use CSQ6CHIP to set channel initiator parameters.

The default parameters for CSQ6CHIP are shown in Table 10. If you want to change any of these values, refer to the detailed descriptions of the parameters.

*Table 10. Default values of CSQ6CHIP parameters*

| Parameter | Description | Default value |
|---|---|---|
| ACTCHL | The maximum number of channels that can be active. | CURRCHL |
| ADAPS | The number of adapter subtasks to use for processing MQI calls. | 8 |
| ADOPTCHK | The elements checked to determine if an MCA should be adopted. | ALL |
| ADOPTMCA | Whether orphaned instances of an MCA are restarted automatically. | NO |
| CURRCHL | The maximum number of channels that can be current. | 200 |
| DISPS | The number of dispatchers to use. | 5 |
| DNSGROUP | The group that the TCP listener should join when registering with Workload Manager for DDNS. | Blank |
| DNSWLM | Whether the TCP listener should register with Workload Manager for DDNS. | NO |
| LSTRTMR | The interval, in seconds, between listener restart attempts. | 60 |
| LUNAME | The name of the LU to use for outbound transmissions. | Blank |
| LUGROUP | The generic LU name of the LU 6.2 listener for the queue-sharing group. | Blank |
| LU62ARM | APPCPMxx SYS1.PARMLIB member name suffix. | Blank |
| LU62CHL | The maximum number of channels that can be current and use the LU 6.2 transmission protocol. | CURRCHL |
| OPORTMAX | The higher end of the range of ports to be used for outgoing channels. | 0 |
| OPORTMIN | The lower end of the range of ports to be used for outgoing channels. | 0 |
| SERVICE | Reserved for use by IBM. | 0 |
| TCPCHL | The maximum number of channels that can be current and use the TCP/IP transmission protocol. | CURRCHL |
| TCPKEEP | Whether the TCP KEEPALIVE facility is to be used or not. | NO |

*Table 10. Default values of CSQ6CHIP parameters (continued)*

| Parameter | Description | Default value |
|---|---|---|
| TCPNAME | The name of the TCP/IP address space or system that will be used. | TCPIP |
| TCPTYPE | TCP/IP interface method. | OESOCKET |
| TRAXSTR | Whether trace should start automatically or not. | YES |
| TRAXTBL | The size of the trace data space in MB. | 2 |

**ACTCHL**

Specifies the maximum number of channels that can be active.

Specify a value in the range 1 through 9999.

The default value is CURRCHL.

**ADAPS**

Specifies the number of adapter subtasks to use for processing MQI calls. As a guideline, the ratio of adapters to dispatchers (the DISPS parameter) should be about 8 to 5. However, if you have only a small number of channels, you do not have to decrease the value of this parameter from the default value.

Specify a value in the range 0 through 9999.

The default value is 8.

Suggested settings:

| | |
|---|---|
| **Test system** | 8 (default) |
| **Production system** | 20 |

Ideally, you should have 20 adapters, which gives greater parallelism of MQI calls. This is particularly important for persistent messages. Fewer adapters might be better for nonpersistent messages.

**ADOPTCHK**

Specifies the elements checked to determine if an MCA should be adopted when a new inbound channel is detected that has the same name as an MCA that is already active.

Specify one of the following:

| | |
|---|---|
| **QMNAME** | Check the queue manager name |
| **NETADDR** | Check the network address |
| **ALL** | Check the queue manager name and network address |
| **NONE** | Do not check any elements |

The default is ALL.

**ADOPTMCA**

Specifies whether an orphaned instance of an MCA should be restarted automatically by the channel initiator when a new inbound channel request matching the ADOPTCHK parameters is detected.

Specify YES or NO.

The default is NO.

**CURRCHL**

Specifies the maximum number of channels that can be current (including server-connection channels with connected clients).

Specify a value in the range 1 through 9999.

The default value is 200.

Suggested settings:

| | |
|---|---|
| **Test system** | 200 (default) |
| **Production system** | 1 000 |

**DISPS**

Specifies the number of dispatchers to use for the channel initiator. As a guideline, allow one dispatcher for each 50 current channels. However, if you have only a small number of channels, you do not have to decrease the value of this parameter from the default value.

If you are using TCP/IP, the greatest number of dispatchers that will be used for TCP/IP channels is 100, even if you specify a larger value here.

Specify a value in the range 1 through 9999.

The default value is 5.

Suggested settings:

| | |
|---|---|
| **Test system** | 5 (default) |
| **Production system** | 20 |

You are recommended to have 20 dispatchers to handle up to 1 000 active channels.

**DNSWLM**

Specifies whether the TCP listener that handles inbound transmissions for the queue-sharing group should register with Workload Manager for Dynamic Domain Name Services.

Specify YES or NO.

The default is NO.

**DNSGROUP**

Specifies the name of the group that the TCP listener that handles inbound transmissions for the queue-sharing group should join when using Workload Manager Dynamic Domain Name Services support.

Specify the name to be used, or blank to use the queue-sharing group name.

The default is blank.

**LSTRTMR**
Specifies the time interval (in seconds) between attempts by MQSeries to restart the listener if there has been an APPC or TCP/IP failure. When the listener is restarted on TCP/IP, it uses the same port and IP address as was used for the original start.

Specify a value in the range 5 through 9999.

The default value is 60.

**Note:** This parameter is ignored if you are using the SOLVE:TCPaccess interface to TCP/IP.

**LUGROUP**
Specifies the generic LU name that the LU 6.2 listener that handles inbound transmissions for the queue-sharing group should use.

Specify the LU name.

The default is blank, which means that this listener cannot be used.

**LUNAME**
Specifies the name of the LU to use for outbound LU 6.2 transmissions. This must be set to the same LU that will be used for inbound transmissions by the listener.

Specify the LU name.

The default is blank, which means that the APPC/MVS default LU should be used; this is variable, so LUNAME should always be set if you are using LU 6.2.

**LU62ARM**
Specifies the suffix of the SYS1.PARMLIB member APPCPMxx, that nominates the LUADD for this channel initiator. The channel initiator issues the OS/390 command `SET APPC=xx` when it starts.

Specify the two-character suffix.

The default is blank which means that no `SET APPC=xx` is issued.

**LU62CHL**
Specifies the maximum number of channels that can be current or clients that can be connected, that use the LU 6.2 transmission protocol. If 0, the LU 6.2 transmission protocol is not used.

Specify a value in the range 0 through 9999.

The default value is CURRCHL.

**OPORTMAX**
Specifies the higher end of the range of port numbers to be used when binding outgoing channels.

Specify a value in the range 0 through 65535. This value must be greater than or equal to the value specified for OPORTMIN.

The default value is 0.

When all the port numbers in the specified range have been used, outgoing channels bind to any available port number. If both OPORTMAX and OPORTMIN are specified as 0, all outgoing channels bind to any available port number.

**Note:** This parameter is not supported if you are using the SOLVE:TCPaccess interface to TCP/IP.

**OPORTMIN**
Specifies the lower end of the range of port numbers to be used when binding outgoing channels.

Specify a value in the range 0 through 65535. This value must be less than or equal to the value specified for OPORTMAX.

The default value is 0.

When all the port numbers in the specified range have been used, outgoing channels bind to any available port number. If both OPORTMAX and OPORTMIN are specified as 0, all outgoing channels bind to any available port number.

**Note:** This parameter is not supported if you are using the SOLVE:TCPaccess interface to TCP/IP.

**SERVICE**
This field is reserved for use by IBM.

**TCPCHL**
Specifies the maximum number of channels that can be current or clients that can be connected, that use the TCP/IP transmission protocol. If 0, the TCP/IP transmission protocol is not used.

The maximum number of TCP/IP sockets used is TCPCHL+DISPS. The OpenEdition MAXFILEPROC parameter (specified in the BPXPRMxx member of SYS1.PARMLIB) controls how many sockets each task is allowed, and thus how many channels each dispatcher is allowed. The number of channels using TCP/IP is limited to MAXFILEPROC*DISPS in this case.

Specify a value in the range 0 through 9999.

The default value is CURRCHL.

**Note:** TCP/IP might not support as many as 9999 channels.

**TCPKEEP**
Specifies whether the TCP KEEPALIVE facility, as specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set, is to be used or not.

Specify YES or NO.

The default is NO.

**TCPNAME**

Specify the name of the TCP/IP system that you are using. This depends on the type of TCP/IP interface that you are using:

| | |
|---|---|
| **IUCV** | The name of the TCP/IP address space. |
| **OpenEdition Sockets** | The name of the OpenEdition stack for TCP/IP, as specified in the SUBFILESYSTYPE NAME parameter in the BPXPRMxx member of SYS1.PARMLIB (described in the *OS/390 OpenEdition Planning* manual). |
| **SOLVE:TCPaccess** | The name of the SOLVE:TCPaccess subsystem. |

The default is TCPIP.

**TCPTYPE**

Specifies the type of TCP/IP interface to be used.

Specify one of the following:

| | |
|---|---|
| **IUCV** | IUCV interface |
| **OESOCKET** | OpenEdition sockets interface |
| **SNSTCPACCESS** | SOLVE:TCPaccess native interface |

The default is OESOCKET.

See Table 11 on page 57 for a summary of the settings.

**TRAXSTR**

Specifies whether trace should start automatically or not.

Specify YES or NO.

The default is YES.

**TRAXTBL**

Specifies the size of the trace data space (in MB).

Specify a value in the range 0 through 2048.

The default value is 2.

**Note:** Whenever you use large OS/390 data spaces, you should ensure that sufficient auxiliary storage is available on your system to support any related OS/390 paging activity. You might also need to increase the size of your SYS1.DUMP data sets.

**Notes:**

1. The channel initiator makes a number of connections to the queue manager that must be allowed for when setting the CTHREAD system parameter (see "Using CSQ6SYSP" on page 31). The number of connections is up to six plus the value of ADAPS plus the value of DISPS.
2. Each dispatcher and each adapter subtask uses a separate OS/390 task. As a guideline, keep the total number of dispatchers and adapter subtasks below 20.

*Table 11. TCP/IP settings*

| Product | Interface | Library | TCPTYPE | TCPNAME |
|---|---|---|---|---|
| IBM® TCP/IP | OpenEdition sockets | SCSQMVR1 | OESOCKET | OpenEdition TCP/IP stack name |
| SOLVE:TCPaccess | OpenEdition sockets | SCSQMVR1 | OESOCKET | OpenEdition TCP/IP stack name |
| SOLVE:TCPaccess | IUCV | SCSQMVR1 | IUCV | TCP/IP address space name |
| SOLVE:TCPaccess | Native SOLVE:TCPaccess | SCSQMVR2 | SNSTCPACCESS | SOLVE:TCPaccess subsystem name |

## Task 18: Set up Batch, TSO, and RRS adapters

- *Repeat this task for each MQSeries queue manager as required.*
- *You might need to perform this task when migrating from a previous version.*

To make the adapters available to batch and TSO applications, add the following MQSeries libraries to the STEPLIB concatenation for your batch application or TSO logon procedure:
- thlqual.SCSQANL*x*
- thlqual.SCSQAUTH

where *x* is the language letter for your national language. (You do not need to do this if the libraries are in the LPA or the link list.)

If the adapter detects an unexpected MQSeries subsystem error, it issues an OS/390 SNAP dump to DDname CSQSNAP, and issues reason code `MQRC_UNEXPECTED_ERROR` to the application. If the CSQSNAP DD statement is not in the application JCL or CSQSNAP is not allocated to a data set under TSO, no dump is taken. If this happens, you could include the CSQSNAP DD statement in the application JCL or allocate CSQSNAP to a data set under TSO and rerun the application. However, because some problems are intermittent, it is recommended that you include a CSQSNAP statement in the application JCL or allocate CSQSNAP to a data set in the TSO logon procedure to capture the reason for failure at the time it occurs.

The supplied program CSQBDEFV improves the portability of your application programs. In CSQBDEFV, you can specify the name of a subsystem to be connected to rather than specifying it in the **MQCONN** or **MQCONNX** call in an application program. You can create a new version of CSQBDEFV for each subsystem. To do this, follow these steps:

1. Copy the MQSeries assembler program CSQBDEFV from thlqual.SCSQASMS to a user library.
2. The supplied program contains the default subsystem name CSQ1. You can retain this name for testing and installation verification. For production subsystems, you can change the NAME=CSQ1 to your one- to four-character subsystem name, or use CSQ1.

   If you are using queue-sharing groups, you can specify a queue-sharing group name instead of CSQ1. If you do this, the program issues a connect request to an active queue manager within that group.
3. Assemble and link-edit the program to produce the CSQBDEFV load module. For the assembly, include the library thlqual.SCSQMACS in your SYSLIB concatenation; use the link-edit parameters `RENT,AMODE=31,RMODE=ANY`. This is shown in the sample JCL in thlqual.SCSQPROC(CSQ4DEFV). Then include the load library in the OS/390 Batch or the TSO JOBLIB or STEPLIB, ahead of thlqual.SCSQAUTH.

## Task 19: Set up the operations and control panels

- *You need only perform this task once.*
- *You might need to perform this task when migrating from a previous version.*

To set up the operations and control panels you must first set up the libraries that contain the required panels, EXECs, messages, and tables. To do this, you must take into account which national language feature is to be used for the panels. When you have done this, you can optionally:

- Update the main ISPF menu for MQSeries operations and control panels
- Change the function key settings

### Setting up the libraries

Follow these steps to install the MQSeries operations and control panels:

1. Include the library thlqual.SCSQEXEC in your SYSEXEC or SYSPROC concatenation. This library, which is allocated with a fixed-block 80 record format during installation, contains the required EXECs.

   **Notes:**

   a. You should put these EXECs into your SYSEXEC concatenation. However, if you want to put them in SYSPROC, it must have a record length of 80 bytes. If you move the EXECs from SYSPROC to SYSEXEC and your system searches SYSPROC before SYSEXEC, you must ensure that the EXECs are deleted from SYSPROC.
   b. Ensure that all the libraries contained in your concatenations are either in the same format (F, FB, V, VB) and have the same block size, or in the order of decreasing block sizes. Otherwise, you might have problems trying to use these panels.
2. | Add thlqual.SCSQAUTH to the TSO logon procedure STEPLIB if it is not in the link list or the LPA.
3. You can either install the MQSeries panel libraries permanently in your ISPF library setup, or allow them to be set up dynamically when the panels are used. For the former choice, you need to do the following:
   a. Include the name of the library containing the operations and control panel definitions in your ISPPLIB concatenation. The name is thlqual.SCSQPNLx, where x is the language letter for your national language.
   b. Install the required tables in your ISPTLIB concatenation. The name is thlqual.SCSQTBLx, where x is the language letter for your national language.
   c. Install the required messages in your ISPMLIB concatenation. The name is thlqual.SCSQMSGx, where x is the language letter for your national language.
   d. Include the library thlqual.SCSQSKL in your ISPSLIB concatenation. This library contains the required skeletons for data set tailoring.
   e. Install the required load modules in your ISPLLIB concatenation. These modules are in the thlqual.SCSQAUTH library with names beginning CSQOX.
4. Test that you can access the MQSeries panels from the TSO Command Processor panel. This is usually option 6 on the ISPF/PDF Primary Options

Menu. The name of the EXEC that you run is CSQOREXX. There are no parameters to specify if you have installed the MQSeries libraries permanently in your ISPF setup as in step 3 on page 59. If you have not, use the following:

```
CSQOREXX thlqual langletter
```

where `langletter` is a letter identifying the national language to be used:

| | |
|---|---|
| **C** | Simplified Chinese |
| **E** | U.S. English (mixed case) |
| **K** | Japanese |
| **U** | U.S. English (uppercase) |

## Updating the ISPF menu

You can update the ISPF main menu to allow access to the MQSeries operations and control panels from ISPF. The required setting for &ZSEL is:

```
CMD(%CSQOREXX thlqual langletter) NEWAPPL(CSQO) PASSLIB
```

For information about `thlqual` and `langletter`, see Step 4 on page 59.

For more details, see the *ISPF Dialog Developer's Guide and Reference* manual.

## Updating the function keys and command settings

You can use the normal ISPF procedures for changing the function keys and command settings used by the panels. The application identifier is CSQO.

However, this is *not* recommended because the help information is not updated to reflect any changes that you have made.

## Task 20: Include the MQSeries dump formatting member

- *Repeat this task for each MQSeries queue manager.*
- *You need to perform this task when migrating from a previous version.*

To be able to format MQSeries dumps using the Interactive Problem Control System (IPCS), copy the data set thlqual.SCSQPROC(CSQ7IPCS) to SYS1.PARMLIB. You should not need to edit this data set.

If you are using OS/390 Version 2.8 or earlier, edit SYS1.PARMLIB member BLSCECTX and add this statement at the end of the member:

```
IMBED MEMBER(CSQ7IPCS) ENVIRONMENT(ALL)
```

(This member is described in the *MVS Initialization and Tuning Reference* manual.)

If you have customized the TSO procedure for IPCS, thlqual.SCSQPROC(CSQ7IPCS) can be copied into any library in the IPCSPARM definition. See the *MVS IPCS Customization* manual for details on IPCSPARM.

You must also include the library thlqual.SCSQPNLA in your ISPPLIB concatenation.

To make the dump formatting programs available to your TSO session or IPCS job, you must also include the library thlqual.SCSQAUTH in your STEPLIB concatenation. (Alternatively SCSQAUTH can be included in the OS/390 LINKLIST concatenation.)

## Task 21: Suppress information messages

- *Repeat this task for each MQSeries queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

If your MQSeries system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the OS/390 console and hardcopy log. The MQSeries-IMS bridge and buffer manager might also produce a large number of information messages.

If required, you can suppress some of these console messages by using the OS/390 message processing facility list, specified by the MPFLSTxx members of SYS1.PARMLIB. The messages you specify still appear on the hardcopy log, but not on the console.

Sample thlqual.SCSQPROC(CSQ4MPFL) shows suggested settings for MPFLSTxx. See the *MVS Initialization and Tuning Reference* manual for more information about MPFLSTxx.

If you want to suppress selected information messages on the hardcopy log, you can use the OS/390 installation exit IEAVMXIT. You can set the following bit switches ON for the required messages:

**CTXTRDTM**
Delete the message.

The message will not be displayed on consoles or logged in hardcopy.

**CTXTESJL**
Suppress from job log.

The message will not go into the JES job log.

**CTXTNWTP**
Do not carry out WTP processing.

The message will not be sent to a TSO terminal or to the system message data set of a batch job.

**Notes:**

1. For full details, refer to the *MVS Installation Exits* book.
2. You are not recommended to suppress messages other than those in the suggested suppression list, CSQ4MPFL.

# Chapter 3. Migrating your queue manager from a previous version

This chapter describes the things that you must consider if you are migrating from a previous version of MQSeries. The following topics are discussed:

- “Migrating to Version 5.2”
- “Reverting to previous versions” on page 70
- “Coexistence with earlier versions of MQSeries” on page 71

After installing the new version, you must IPL the system so that the new MQSeries early code is brought into use.

## Migrating to Version 5.2

When you migrate from a previous version of MQSeries for OS/390, you can continue to use your existing subsystems with the new version, including their page sets, log data sets, object definitions, and initialization input data sets. You can continue to use your existing queues, including system queues such as the SYSTEM.CHANNEL.SYNCQ. **You should not cold start your queue managers when migrating from a previous version because you do not need to. If you do, you will lose all your messages and other information such as channel state.**

However, there are some tasks that you need to perform when migrating from a previous version. Whether you need to perform each task depends on which of the new features you want to use, and which level of MQSeries you are migrating from. Generally, you need to perform more tasks if you are migrating from a version of MQSeries that was not the previous version (that is, Version 1.2 or earlier); however, you do not need to install the intervening versions.

“What’s new for this release” on page xii describes the new functions for this release. Consider which of these functions you want to use before customizing MQSeries because you might not need to perform all the migration tasks.

If you are migrating from Version 2.1 to Version 5.2, you must consider the tasks in this section. If you are migrating from a previous version of MQSeries for OS/390, you must also consider the points discussed in the following sections:

- “Additional steps when migrating from Version 1.2” on page 66
- “Additional steps when migrating from Version 1.1.4” on page 69

**Software levels**
: The minimum levels for some of the items of software required to use MQSeries have changed. Check that you have the correct levels of prerequisite and corequisite software from the list in the *MQSeries for OS/390 Concepts and Planning Guide*.

**System parameter module**
: There are several new system parameters (QSGDATA, RESAUDIT, DEALLCT, and UNIT2) and the MAXRTU parameter supersedes the MAXALLC parameter, which is no longer used. There are also several changed parameters. These are described in “What’s new for this release” on page xii. Consider whether you need to use these parameters and change your system parameter module accordingly.

If you do not need to use these parameters, you do not need to relinkedit your system parameter module.

**Channel initiator parameter module**
There are several new channel initiator parameters (ADOPTCHK, ADOPTMCA, DNSGROUP, DNSWLM, LUGROUP, OPORTMAX, and OPORTMIN). These are described in "What's new for this release" on page xii. Consider whether you need to use these parameters and change your channel initiator parameter module accordingly.

If you do not need to use these parameters, you do not need to relinkedit your channel initiator parameter module.

**Initialization data sets**
A new sample input initialization data set for queue sharing groups called thlqual.SCSQPROC(CSQ4INSS) is supplied with MQSeries. If you are planning to use queue-sharing groups, customize and include this data set.

Review sample data set thlqual.SCSQPROC(CSQ5INYG) to see if you want to use the new default buffer pool, storage class, and page set definitions.

**Logs** Review sample data set thlqual.SCSQPROC(CSQ4BSDS) to see if you want to use the new default settings for log placement and size.

If you are using very large messages, the amount of storage required for your log and archive data sets might increase. This is described in the *MQSeries for OS/390 Concepts and Planning Guide*.

**Installation verification program**
The name of the samples for the IVP in thlqual.SCSQPROC have been changed to CSQ4IVPQ and CSQ4IVPR. New samples called CSQ4IVPG and CSQ4IVPS have been added for the queue-sharing group IVP. These are described in "Chapter 4. Testing your queue manager" on page 73.

**Library names**
Change any MQSeries library names in all STEPLIBs if they have new names.

**Migrating queues and queue definitions to shared queues**
*MQSeries for OS/390 System Administration Guide* describes how to migrate your existing queues and queue definitions to be used as shared queues. You do not have to do this, but you should consider it if you are going to use shared queues.

**Change log inventory utility (CSQJU003)**
The STARTRBA and ENDRBA keyword value of NEWLOG must end in 000 and FFF respectively.

**Return codes**
MQRC_PAGESET_FULL and new return code MQRC_STORAGE_MEDIUM_FULL have the same value.

**Data conversion exits**
Data conversion exits written for MQSeries for OS/390 Version 2.1 will continue to function correctly with Version 5.2. However, they are unable to convert messages containing text using the Unicode UCS-2 coded character sets (1200, 13488, 17584) and need to be updated to do so, if you require such conversions.

Exits generated using the CSQUCVX utility need to be reassembled and link-edited, ensuring that you use the thlqual.SCSQMACS library supplied with Version 5.2. See the *MQSeries Application Programming Guide* and the CSQ4BAX9 and CSQ4CAX9 samples for information about using CSQUCVX.

Other exits place calls to **MQXCNVC** to perform data conversion. If these exits need to deal with UCS-2, the *Options* parameter of this call must be updated to specify the byte order of the UCS-2 text. See the *MQSeries Application Programming Reference* manual for information about **MQXCNVC** and sample exit CSQ4BAX8, which demonstrates how to calculate this parameter.

If these exits are not updated, applications will not be able to convert to or from Unicode UCS-2 CCSIDs. Typically, this is seen in the response from **MQGET**, which returns the message unconverted with reason code MQRC_SOURCE_INTEGER_ENC_ERROR or MQRC_TARGET_INTEGER_ENC_ERROR (this depends on the behavior of the exit).

**Note:** Early versions of the CSQ4BAX8 sample exit incorrectly filled in the *Options* parameter of **MQXCNVC**, and exits that have copied this behavior might convert UCS-2 text incorrectly, without reporting a failure.

## Additional steps when migrating from Version 1.2

If you intend to migrate from Version 1.2 to Version 5.2, you need to consider the following when you customize your new version, in addition to the tasks in the previous section (you don't need to install and customize the intervening versions):

**Software levels**
The minimum levels for many of the items of software required to use MQSeries for OS/390 have changed (OS/390, CICS, and IMS in particular). Check that you have the correct levels for your prerequisite and corequisite software from the list in the *MQSeries for OS/390 Concepts and Planning Guide*.

**System parameter module**
There are new system parameters (EXITLIM, EXITTCB, and WLMTIME) and channel initiator parameters (TCPTYPE, LU62ARM, and LSTRTMR). Consider whether you need to use these new parameters, and change your parameter modules again accordingly. See "Task 16: Tailor your system parameter module" on page 30 and "Task 17: Tailor the channel initiator parameter module" on page 50 for more information.

**Installation process**
There are several changes to the installation process, and some new libraries. The two distributed queuing features for the non-CICS mover have been incorporated into the base product, and the CICS mover has been made an optional feature. The CICS bridge has also been incorporated into the base product.

These are described in the *MQSeries for OS/390 Program Directory*.

**Automatic Restart Manager (ARM)**
The OS/390 Automatic Restart Manager (ARM) is now supported. This support coexists on the same OS/390 image with earlier releases that do not support ARM. The queue managers and channel initiators in the earlier releases do not register with ARM and so can not be restarted automatically.

If you do not want to use ARM with your Version 5.2 queue managers and channel initiators, specify RESTART_ATTEMPTS(0) for the MQSeries element in your ARM policy. Note that if you do not specify MQSeries elements in your ARM policy, default ARM policies are used for MQSeries.

OS/390 ARM support is described in the *MQSeries for OS/390 System Administration Guide*

**Clusters**
MQSeries now supports clustering. Before you use clustering you must review all of your applications to determine whether each one can operate in a clustering environment. You might have to modify your applications to remove or manage inter-message affinity. Applications that attempt to open non-existent queues might experience delays, or might even successfully open a queue somewhere in the cluster.

You also need to create the new system objects required for clustering. These are described in the *MQSeries for OS/390 Concepts and Planning Guide*.

There is a cluster workload user exit; if you use this you need to add a CSQXLIB DD statement to your queue manager started task procedure, *xxxx*MSTR, and ensure that you have access to the LE run-time library SCEERUN.

Cluster support is described in the *MQSeries Queue Manager Clusters* manual.

**Storage classes**

The supplied default for storage class SYSTEM (which was used by many of the SYSTEM queues) has been changed to page set 01, so that messages are not put on page set 00.

If you currently use the defaults supplied, this change will probably have no effect, even if you use the DEFINE REPLACE option for your storage class definitions in your initialization input data set. This is because some of the queues using that storage class (like the SYSTEM.CHANNEL.SYNCQ for example) have messages on them permanently. If you want to move the queues to another page set, follow the procedure given in the *MQSeries for OS/390 System Administration Guide*.

**Initialization data sets**

The sample input initialization data sets supplied with MQSeries have been reorganized and renamed. This is described in "Task 12: Customize the initialization input data sets" on page 24.

**Resource Recovery Services (RRS)**

You can migrate your existing batch/TSO MQSeries applications to exploit RRS coordination with little or no application program change. If you link-edit your MQSeries application with the CSQBRRSI adapter, **MQCMIT** and **MQBACK** will synchronize your unit of work across MQSeries and all other RRS-enabled resource managers. If you link-edit your MQSeries application with the CSQBRSTB adapter, you must change **MQCMIT** and **MQBACK** to **SRRCMIT** and **SRRBACK**.

Version 5.2 continues to support the non-RRS managed batch adapter in addition to supporting the RRS managed adapter. Thus different versions of MQSeries queue managers can coexist on the same OS/390 image.

**OpenEdition sockets**

OpenEdition sockets are now available for use as an alternative to IUCV. If you are using OS/390 Version 2.5 or later, and are using IBM TCP/IP for distributed queuing, IUCV is not available. You must set the TCPTYPE channel initiator parameter to OESOCKETS (as described in Table 10 on page 51). Using OpenEdition sockets, you do not need to restart the channel initiator if TCP/IP has to be restarted.

**Channel initiator security**

Channel initiator user ID checking has been changed and some new facilities added. See "User IDs used by the channel initiator" on page 185 for details, and review your channel definitions to ensure that you are getting the security control you want.

**Channel initiator snap dumps**

The channel initiator can now record error information in a data set instead of taking a dump. Add the CSQSNAP DD statement to your channel initiator started task procedure to support this.

**IMS language interface module**

The IMS language interface module CSQ2LI00 is no longer supported. All IMS applications should use the IMS supplied DFSLI000 module.

**Euro currency symbol**

Support for the new euro currency symbol has been added to MQSeries. If you need to modify your applications to use this symbol, ensure that they use one of the coded character sets that include it. These are described in the *MQSeries Application Programming Reference* manual. If you need to change the coded character set used by your queue manager, use the CCSID parameter of the system parameter module. This is described in “Using CSQ6SYSP” on page 31.

**Queue object size**

The size of queue objects has increased for Version 5.2 to allow for the new cluster attributes. MQSeries automatically updates each of these queue objects the first time that it is changed. This happens whether you are using clustering or not. However, due to the nature of space reclamation in MQSeries, the space used on page set zero might increase dramatically until all these queue objects have been updated.

To avoid this, you should run a job similar to that in Figure 4 that changes all of your queue objects, enabling MQSeries to update them all at the same time.

If you do not run a job like this, applications attempting to open queues might receive return code MQRC_OBJECT_IN_USE. This includes attempts by the channel initiator to open transmission queues.

```
//STEP1    EXEC  PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB  DD    DISP=SHR,DSN=thlqual.SCSQANLE
//         DD    DISP=SHR,DSN=thlqual.SCSQAUTH
//OUTPUT1  DD    DISP=OLD,DSN=MY.MQSERIES.COMMANDS(DEFS)
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(CMDINP) MAKEDEF(OUTPUT1)
/*
//CMDINP   DD *
DISPLAY QUEUE(*) TYPE(QLOCAL) ALL
DISPLAY QUEUE(*) TYPE(QMODEL) ALL
DISPLAY QUEUE(*) TYPE(QALIAS) ALL
DISPLAY QUEUE(*) TYPE(QREMOTE) ALL
/*
//*   STEP2
//*********************************************************************
//* PERFORM A GLOBAL CHANGE ON THE OUTPUT DATA SET FROM STEP 1, THAT    *
//* IS: MY.MQSERIES.COMMANDS(DEFS).  CHANGE 'NOREPLACE' TO 'REPLACE'    *
//* THE CHANGED MY.MQSERIES.COMMANDS(DEFS) WILL BE THE INPUT FOR STEP3. *
//*********************************************************************
//*
//STEP3    EXEC  PGM=CSQUTIL,PARM='CSQ1'
//STEPLIB  DD    DISP=SHR,DSN=thlqual.SCSQANLE
//         DD    DISP=SHR,DSN=thlqual.SCSQAUTH
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
COMMAND DDNAME(DEFINES)
/*
//*********************************************************************
//* THE DEFINE COMMAND FOR THE SYSTEM.COMMAND.INPUT QUEUE MIGHT FAIL,   *
//* BUT THIS DOES NOT MATTER. ALTERNATIVELY, REMOVE THE DEFINE FOR THAT *
//* QUEUE FROM THE OBJECT DATA SET FROM STEP 1.                         *
//*********************************************************************
//*
//DEFINES  DD DISP=SHR,DSN=MY.MQSERIES.COMMANDS(DEFS)
```

*Figure 4. Example job for migrating queue objects*

## Additional steps when migrating from Version 1.1.4

If you intend to migrate from Version 1.1.4 to Version 5.2, you need to consider the following when you customize your new version in addition to the tasks in the previous sections (you don't need to install and customize the intervening versions):

**Index queues**
There is a new queue attribute, INDXTYPE, that allows the queue manager to expedite **MQGET** operations on the queue. To improve channel performance, you should set INDXTYPE to MSGID for the SYSTEM.CHANNEL.SYNCQ queue. In general, you will then need to restart the queue manager to have this take effect. See the *MQSeries MQSC Command Reference* manual for more information.

**Channel attributes**
There is a new channel attribute, NPMSPEED, that specifies a class of service for nonpersistent messages on the channel. **The class applied to existing channel definitions is FAST, which means that nonpersistent messages might be lost if there is a channel error.** If this is not acceptable, you must alter your channel definitions to have NPMSPEED(NORMAL).

There are also two other new channel attributes, HBINT and BATCHINT. See the *MQSeries MQSC Command Reference* manual for more information.

**Distributed queuing**
There are new libraries for distributed queuing: thlqual.SCSQMVR1 if you are using the OpenEdition sockets or IUCV interface and thlqual.SCSQMVR2 if you are using the SOLVE:TCPaccess interface. Add the appropriate library to the STEPLIB DD statement of the JCL used for your channel initiator started task procedures (xxxxCHIN) and data conversion utility (CSQUCVX).

**C/370** Use of the C/370 library product by MQSeries is no longer supported (although MQSeries applications can still use it). You must now use LE instead. Change your channel initiator started-task procedures (xxxxCHIN) and data conversion utility (CSQUCVX) JCL accordingly.

**TCP/IP Version 2**
TCP/IP Version 2 is no longer supported.

## Reverting to previous versions

This section tells you what to do if you want to revert to using a previous version of MQSeries for OS/390 for some exceptional reason.

If you choose to revert to a previous version of MQSeries for OS/390, note that, as a general rule, data such as attributes (or in some cases, objects) relating to the new function in Version 5.2 will be lost. The more you have used the new functions of Version 5.2, the less practical it will be to go back to an earlier version.

If you want to revert to an earlier version of MQSeries, contact your IBM support center. The support center will provide a PTF that you must apply to your system in order to do this. The information in the PTF includes a description of the data relating to new function that will be lost.

When you have reverted to an earlier version, shared queues and their messages will not be accessible to the queue manager. They remain on the Coupling Facility and in DB2 for other members of the queue-sharing group, or awaiting forward migration. Your DB2, Coupling Facility and queue-sharing group setup will not be lost.

## Coexistence with earlier versions of MQSeries

This topic discusses coexistence issues for the following:
- “Multiple queue manager versions”
- “Operations and control panels”
- “Application stubs”

### Multiple queue manager versions

There can be several MQSeries subsystems in an OS/390 image, and they can use different versions of MQSeries, provided the MQSeries early code modules are of the latest version being used. (These modules are loaded at OS/390 IPL time and are shared among all the MQSeries subsystems in the OS/390 image.)

This means that you can run one queue manager with Version 5.2 and another in the same image with Version 2.1 or 1.2, provided that the early code is that of Version 5.2.

| Use STEPLIBs to control which level of MQSeries is used.

ARM support and RRS support allow such coexistence, as explained in “Additional steps when migrating from Version 1.2” on page 66.

### Operations and control panels

When using the operations and control panels, the MQSeries libraries you use in ISPF must match those of the queue manager you are working with. That is, the panels at the Version 5.2 level will work only with a Version 5.2 queue manager, and Version 2.1 panels will work only with a Version 2.1 queue manager, and so on.

### Application stubs

The stub modules that are link-edited with applications and exits (CSQASTUB, CSQBRSSI, CSQBRSTB, CSQBSTUB, CSQCSTUB, CSQQSTUB, and CSQXSTUB) might not work with earlier versions of the queue manager. For example, stubs supplied with Version 2.1 can be used by applications running on a Version 2.1 or 5.2 queue manager; however, if the application is run on a Version 1.2 queue manager, it might not work, or might end abnormally.

# Chapter 4. Testing your queue manager

When you have customized or migrated your queue manager, you can test it by running some of the sample applications shipped with MQSeries.

You can then compile and link-edit whichever of the other samples are appropriate to your installation using the sample JCL supplied.

This chapter tells you about:

- "Running the basic installation verification program"
- "Testing for queue-sharing groups" on page 77
- "Testing for distributed queuing" on page 79
- "Testing for C, C++, COBOL, PL/I, and CICS" on page 82

## Running the basic installation verification program

After you have installed and customized MQSeries, you can use the supplied installation verification program, CSQ4IVP1, to confirm that MQSeries is operational. This is a batch assembler IVP that verifies the base MQSeries without using the C, COBOL, or CICS samples.

The Batch Assembler IVP is link-edited by SMP/E and the load modules are shipped in library thlqual.SCSQLOAD.

After you have completed both the SMP/E APPLY step and the customization steps, run the Batch Assembler IVP.

### Overview of the CSQ4IVP1 application

CSQ4IVP1 is a batch application that connects to your MQSeries subsystem and performs these basic functions:

- Issues MQI calls
- Communicates with the command server
- Verifies triggering is active
- Generates and deletes a dynamic queue

### Preparing to run CSQ4IVP1

Before you run CSQ4IVP1:

1. Check that the IVP entries are in the CSQINP2 data set concatenation in the MQSeries startup program. The IVP entries are supplied in member thlqual.SCSQPROC(CSQ4IVPQ). If not, add the definitions supplied in thlqual.SCSQPROC(CSQ4IVPQ) to your CSQINP2 concatenation. If MQSeries is currently running, you will need to restart it so that these definitions can take effect.
2. The sample JCL, CSQ4IVPR, required to run the installation verification program is in library thlqual.SCSQPROC.

   Customize the CSQ4IVPR JCL with the high-level qualifier for the MQSeries libraries, the national language you want to use, the four-character MQSeries queue manager name, and the destination for the job output.

3. Update RACF to allow CSQ4IVP1 to access its resources if MQSeries security is active.

   To run CSQ4IVP1 when MQSeries security is enabled, you need a RACF user ID with authority to access the objects. For details of defining resources to RACF, see “Part 5. Setting up security” on page 135. The user ID that runs the IVP must have the following access authority:

| **Authority** | **Profile** | **Class** |
|---|---|---|
| READ | ssid.DISPLAY.PROCESS | MQCMDS |
| UPDATE | ssid.SYSTEM.COMMAND.INPUT | MQQUEUE |
| UPDATE | ssid.SYSTEM.COMMAND.REPLY.MODEL | MQQUEUE |
| UPDATE | ssid.CSQ4IVP1.** | MQQUEUE |
| READ | ssid.BATCH | MQCONN |

   These requirements assume that all MQSeries security is active. The RACF commands to activate MQSeries security are shown in Figure 5. This example assumes that the MQSeries subsystem name is CSQ1 and that the user ID of the person running sample CSQ4IVP1 is TS101.

```
RDEFINE MQCMDS CSQ1.DISPLAY.PROCESS
PERMIT CSQ1.DISPLAY.PROCESS CLASS(MQCMDS) ID(TS101) ACCESS(READ)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.INPUT
PERMIT CSQ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.REPLY.MODEL
PERMIT CSQ1.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.CSQ4IVP1.**
PERMIT CSQ1.CSQ4IVP1.** CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQCONN CSQ1.BATCH
PERMIT CSQ1.BATCH CLASS(MQCONN) ID(TS101) ACCESS(READ)
```

*Figure 5. RACF commands for CSQ4IVP1*

## Running CSQ4IVP1

When you have completed these steps, start your MQSeries subsystem. If MQSeries is already running and you have made changes to CSQINP2, you must stop MQSeries and restart it.

The IVP runs as a batch job. Customize the job card to meet the submission requirements of your installation.

## Checking the results of CSQ4IVP1

The IVP is split into eight stages; each stage must complete with a zero completion code before the next stage is run. The IVP generates a report, listing:

- The name of queue manager that is being connected to.
- A one-line message showing the completion code and the reason code returned from each stage.

For an explanation of the completion and reason codes, see the *MQSeries for OS/390 Messages and Codes* manual.

Some stages have more than one MQI call and, in the event of failure, a message is issued indicating the specific MQI call that returned the failure. Also, for some stages the IVP puts explanatory and diagnostic information into a comment field.

The IVP job requests exclusive control of certain queue manager objects and therefore should be single threaded through the system. However, there is no limit to the number of times the IVP can be run against your MQSeries subsystem.

The functions performed by each stage are:

**Stage 1**
Connect to the queue manager by issuing **MQCONN**.

**Stage 2**
Determine the name of the system-command input queue used by the command server to retrieve request messages. This queue receives display requests from Stage 5.

To do this, the sequence of calls is:

1. Issue an **MQOPEN**, specifying the queue manager name, to open the queue manager object.
2. Issue an **MQINQ** to find out the name of the system-command input queue.
3. Issue an **MQCLOSE** to close the queue manager object.

On successful completion of this stage, the name of the system-command input queue is displayed in the comment field.

**Stage 3**
Open an initiation queue using **MQOPEN**.

This queue is opened at this stage in anticipation of a trigger message, which arrives as a result of the command server replying to the request from Stage 5. The queue must be opened for input to meet the triggering criteria.

**Stage 4**
Create a permanent dynamic queue using the CSQ4IVP1.MODEL queue as a model. The dynamic queue has the same attributes as the model from which it was created. This means that when the replies from the command server request in Stage 5 are written to this queue, a trigger message is written to the initiation queue opened in Stage 3.

Upon successful completion of this stage, the name of the permanent dynamic queue is indicated in the comment field.

**Stage 5**
Issue an **MQPUT1** request to the command server command queue.

A message of MQMT_REQUEST is written to the system-command input queue requesting a display of process CSQ4IVP1. The message descriptor for the message specifies the permanent dynamic queue created in Stage 4 as the reply-to queue for the command server's response.

**Stage 6**
Issue an **MQGET** request from the initiation queue. At this stage, a GET WAIT with an interval of one minute is issued against the initiation queue opened in Stage 3. The message returned is expected to be the trigger message generated by the command server's response messages being written to the reply-to queue.

**Stage 7**
Delete the permanent dynamic queue created in Stage 4. As the queue still has messages on it, the MQCO_PURGE_DELETE option is used.

**Stage 8**
Disconnect from the queue manager using **MQDISC**.

After running the IVP, you can delete any objects that you no longer require.

If the IVP does not run successfully, try each step manually to find out which function is failing.

## Testing for queue-sharing groups

The basic installation verification program tests non-shared queues. It can be used whether the queue manager is a member of a queue-sharing group or not. After running the basic IVP, you can test for shared queues by using the CSQ4IVP1 installation verification program with different queues. This will also test that DB2 and the Coupling Facility are set up correctly.

### Preparing to run CSQ4IVP1 for a queue-sharing group

Before you run CSQ4IVP1:

1. Add the Coupling Facility structure that the IVP uses to your CFRM policy data set, as described in “Task 10: Set up the Coupling Facility” on page 22. The supplied samples use a structure called APPLICATION1, but you can change this if you want.
2. Check that the IVP entries are in the CSQINP2 data set concatenation in the MQSeries startup program. The IVP entries are supplied in member thlqual.SCSQPROC(CSQ4IVPG). If they are not, add the definitions supplied in thlqual.SCSQPROC(CSQ4IVPG) to your CSQINP2 concatenation. If MQSeries is currently running, you will need to restart it so that these definitions can take effect.
3. Change the name of the Coupling Facility structure used in thlqual.SCSQPROC(CSQ4IVPG) if necessary.
4. The sample JCL, CSQ4IVPS, required to run the installation verification program for a queue-sharing group is in library thlqual.SCSQPROC.

   Customize the CSQ4IVPS JCL with the high-level qualifier for the MQSeries libraries, the national language you want to use, the four-character MQSeries queue manager name, and the destination for the job output.
5. Update RACF to allow CSQ4IVP1 to access its resources if MQSeries security is active.

   To run CSQ4IVP1 when MQSeries security is enabled, you need a RACF user ID with authority to access the objects. For details of defining resources to RACF, see “Part 5. Setting up security” on page 135. The user ID that runs the IVP must have the following access authority in addition to that required to run the basic IVP:

   | Authority | Profile | Class |
   |---|---|---|
   | UPDATE | ssid.CSQ4IVPG.** | MQQUEUE |

   These requirements assume that all MQSeries security is active. The RACF commands to activate MQSeries security are shown in Figure 6. This example assumes that the MQSeries subsystem name is CSQ1 and that the user ID of the person running sample CSQ4IVP1 is TS101.

```
RDEFINE MQQUEUE CSQ1.CSQ4IVPG.**
PERMIT CSQ1.CSQ4IVPG.** CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)
```

*Figure 6. RACF commands for CSQ4IVP1 for a queue-sharing group*

## Running CSQ4IVP1 for a queue-sharing group

When you have completed these steps, start your MQSeries subsystem. If MQSeries is already running and you have made changes to CSQINP2, you must stop MQSeries and restart it.

The IVP runs as a batch job. Customize the job card to meet the submission requirements of your installation.

## Checking the results of CSQ4IVP1 for a queue-sharing group

The IVP for queue-sharing groups works in the same way as the basic IVP, except that the queues that are created are called CSQIVPG.*xx*. Follow the instructions given in “Checking the results of CSQ4IVP1” on page 75 to check the results of the IVP for queue-sharing groups.

## Testing for distributed queuing

You can use the supplied installation verification program, CSQ4IVPX, to confirm that distributed queuing (without CICS) is operational.

### Overview of CSQ4IVPX job

CSQ4IVPX is a batch job that starts the channel initiator and issues the DISPLAY DQM MQSC command. This verifies that all major aspects of distributed queuing are operational, while avoiding the need to set up channel and network definitions.

### Preparing to run CSQ4IVPX

Before you run CSQ4IVPX:

1. The sample JCL, CSQ4IVPX, required to run the installation verification program is in library thlqual.SCSQPROC.

   Customize the CSQ4IVPX JCL with the high-level qualifier for the MQSeries libraries, the national language you want to use, the four-character MQSeries subsystem name, and the destination for the job output. If you have a customized channel initiator parameter module, replace CSQXPARM with the name of your module.
2. Update RACF to allow CSQ4IVPX to access its resources if MQSeries security is active. To run CSQ4IVPX when MQSeries security is enabled, you need a RACF user ID with authority to access the objects. For details of defining resources to RACF, see “Part 5. Setting up security” on page 135. The user ID that runs the IVP must have the following access authority:

   | Authority | Profile | Class |
   |---|---|---|
   | CONTROL | ssid.START.CHINIT and ssid.STOP.CHINIT | MQCMDS |
   | UPDATE | ssid.SYSTEM.COMMAND.INPUT | MQQUEUE |
   | UPDATE | ssid.SYSTEM.CSQUTIL.* | MQQUEUE |
   | READ | ssid.BATCH | MQCONN |
   | READ | ssid.DISPLAY.DQM | MQCMDS |

   These requirements assume that the connection security profile ssid.CHIN has been defined (as shown in “Connection security profiles for distributed queuing” on page 149), and that all MQSeries security is active. The RACF commands to do this are shown in Figure 7 on page 80. This example assumes that:
   - The MQSeries subsystem name is CSQ1
   - The user ID of the person running sample CSQ4IVPX is TS101
   - The channel initiator address space is running under the user ID CSQ1MSTR
3. Update RACF to allow the channel initiator address space the following access authority:

   | Authority | Profile | Class |
   |---|---|---|
   | READ | ssid.CHIN | MQCONN |
   | UPDATE | ssid.SYSTEM.COMMAND.INPUT | MQQUEUE |
   | UPDATE | ssid.SYSTEM.CHANNEL.INITQ | MQQUEUE |
   | UPDATE | ssid.SYSTEM.CHANNEL.SYNCQ | MQQUEUE |
   | ALTER | ssid.SYSTEM.CLUSTER.COMMAND.QUEUE | MQQUEUE |

| Authority | Profile | Class |
|---|---|---|
| UPDATE | ssid.SYSTEM.CLUSTER.TRANSMIT.QUEUE | MQQUEUE |
| ALTER | ssid.SYSTEM.CLUSTER.REPOSITORY.QUEUE | MQQUEUE |
| CONTROL | ssid.CONTEXT | MQADMIN |

The RACF commands to do this are also shown in Figure 7.

```
RDEFINE MQCMDS CSQ1.DISPLAY.DQM
PERMIT CSQ1.DISPLAY.DQM CLASS(MQCMDS) ID(TS101) ACCESS(READ)

RDEFINE MQCMDS CSQ1.START.CHINIT
PERMIT CSQ1.START.CHINIT CLASS(MQCMDS) ID(TS101) ACCESS(CONTROL)

RDEFINE MQCMDS CSQ1.STOP.CHINIT
PERMIT CSQ1.STOP.CHINIT CLASS(MQCMDS) ID(TS101) ACCESS(CONTROL)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.INPUT
PERMIT CSQ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ID(TS101,CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CSQUTIL.*
PERMIT CSQ1.SYSTEM.CSQUTIL.* CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQCONN CSQ1.BATCH
PERMIT CSQ1.BATCH CLASS(MQCONN) ID(TS101) ACCESS(READ)

RDEFINE MQCONN CSQ1.CHIN
PERMIT CSQ1.CHIN CLASS(MQCONN) ID(CSQ1MSTR) ACCESS(READ)

RDEFINE MQQUEUE CSQ1.SYSTEM.CHANNEL.SYNCQ
PERMIT CSQ1.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.COMMAND.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(ALTER)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.TRANSMIT.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.REPOSITORY.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(ALTER)

RDEFINE MQQUEUE CSQ1.SYSTEM.CHANNEL.INITQ
PERMIT CSQ1.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQADMIN CSQ1.CONTEXT
PERMIT CSQ1.CONTEXT CLASS(MQADMIN) ID(CSQ1MSTR) ACCESS(CONTROL)
```

*Figure 7. RACF commands for CSQ4IVPX*

## Running CSQ4IVPX

When you have completed these steps, start your MQSeries subsystem.

The IVP runs as a batch job. Customize the job card to meet the submission requirements of your installation.

## Checking the results of CSQ4IVPX

CSQ4IVPX runs the CSQUTIL MQSeries utility to issue three MQSC commands. The SYSPRINT output data set should look like Figure 8, although details might differ depending on your channel initiator parameters.

- You should see the commands **(1)** each followed by several messages.
- The last message from each command should be “CSQ9022I ... NORMAL COMPLETION” **(2)**.
- The job as a whole should complete with return code 0 **(3)**.

```
 CSQU000I CSQUTIL IBM MQSeries for OS/390 - V5.2
 CSQU001I CSQUTIL Queue Manager Utility - 2000-05-09 09:06:48
 COMMAND
 CSQU127I CSQUTIL Executing COMMAND using input from CSQUCMD data set
 CSQU055I CSQUTIL Target queue manager is CSQ1
 CSQU120I CSQUTIL Connecting to queue manager CSQ1
 CSQU121I CSQUTIL Connected to queue manager CSQ1
  START CHINIT PARM(CSQXPARM)
(1)
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
CSQM138I +CSQ1 CSQMSCHI CHANNEL INITIATOR STARTING
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQXCRPS ' START CHINIT' NORMAL COMPLETION
(2)
  DISPLAY DQM
(1)
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
CSQM137I +CSQ1 CSQMDDQM  DISPLAY DQM COMMAND ACCEPTED
CSQN205I   COUNT=      12, RETURN=00000000, REASON=00000000
CSQX830I +CSQ1 CSQXRDQM Channel initiator active
CSQX845I +CSQ1 CSQXRDQM TCP/IP address space name is TCPIP
CSQX848I +CSQ1 CSQXRDQM TCP/IP listener not started
CSQX849I +CSQ1 CSQXRDQM LU 6.2 listener not started
CSQX832I +CSQ1 CSQXRDQM 5 dispatchers started, 5 requested
CSQX831I +CSQ1 CSQXRDQM 8 adapter subtasks started, 8 requested
CSQX840I +CSQ1 CSQXRDQM 0 channel connections current, maximum 200
CSQX841I +CSQ1 CSQXRDQM 0 channel connections active, maximum 200
CSQX842I +CSQ1 CSQXRDQM 0 channel connections starting,
0 stopped, 0 retrying
CSQ9022I +CSQ1 CSQXCRPS ' DISPLAY DQM' NORMAL COMPLETION
(2)
  STOP  CHINIT
(1)
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000004
CSQM137I +CSQ1 CSQMTCHI  STOP CHINIT COMMAND ACCEPTED
CSQN205I   COUNT=       2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQXCRPS ' STOP CHINIT' NORMAL COMPLETION
(2)
 CSQU057I CSQUCMDS 3 commands read
 CSQU058I CSQUCMDS 3 commands issued and responses received
 CSQU143I CSQUTIL 1 COMMAND statements attempted
 CSQU144I CSQUTIL 1 COMMAND statements executed successfully
 CSQU148I CSQUTIL Utility completed, return code=0
(3)
```

*Figure 8. Example output from CSQ4IVPX*

## Testing for C, C++, COBOL, PL/I, and CICS

You can test for C, C++, COBOL, PL/I, or CICS, using the sample applications supplied with MQSeries. Although the IVP (CSQ4IVP1) is supplied as a load module, the samples are supplied as source modules.

For more information about sample applications, see the *MQSeries Application Programming Reference* and *MQSeries Using C++* manuals.

# Part 2. Customizing for CICS

# Chapter 5. Setting up the CICS adapter

This chapter tells you how to make the MQSeries-CICS adapter (generally referred to in this book as the CICS adapter) available to your CICS subsystem. If you are not familiar with defining resources to CICS, refer to:

- The *CICS System Definition Guide* for general information on setting up a CICS subsystem.
- The *CICS Resource Definition Guide*, for background information on defining resources to CICS, details of and the command syntax of the CEDA transaction, and the MIGRATE command.
- The *CICS Operations and Utilities Guide* and the *CICS Resource Definition Guide* for details of the CSD utility program (DFHCSDUP).

## Resource definition

This section takes you through the steps you must perform to define the resources for the CICS adapter.

### Updating the CSD

This section describes the updates required for the CICS system definition (CSD) data set for the CICS adapter. It also describes the CSD updates required for the distributed queuing facility (if you want to use the “CICS mover”) and the CICS sample application programs. However, it does not contain all the information required to complete these tasks. If you are implementing distributed queuing, see “Appendix C. Enabling distributed queuing using CICS ISC” on page 241. If you intend to use the CICS sample application programs, see the *MQSeries Application Programming Guide*.

You must use resource definition online (RDO) to add new groups to the CSD data set. The new groups must contain definitions of:
- The supplied adapter programs
- The supplied adapter management transactions
- The supplied sets of BMS maps, required for the adapter panels

To update the CSD, run the CICS offline utility program, DFHCSDUP, with the supplied sample input data sets:
- thlqual.SCSQPROC(CSQ4B100)
- thlqual.SCSQPROC(CSQ4D100)
- thlqual.SCSQPROC(CSQ4S100)

Where:

| **This data set...** | **Provides the definitions required for...** |
|---|---|
| CSQ4B100 | CICS adapter |
| CSQ4D100 | Distributed queuing using CICS ISC (this is optional) |
| CSQ4S100 | Supplied samples |

### Setting up the CICS adapter

Each of these data sets contains sample CICS definitions that must be tailored. To preserve the originals, copy these data sets into a user JCL library whose name contains the MQSeries subsystem name, for example, MQS.CSQ1.USERJCL, and tailor them there.

**Note:** With some versions of CICS, you might receive warning messages about obsolete keywords; you can ignore these.

Ensure that any user-written CICS applications that issue MQI calls, and the resources they use, are also defined to the CSD. You can edit the input data set, to include definitions of user-programs and their resources.

You can add this fragment of JCL to your CSD upgrade (DFHCSDUP) job to define the MQSeries supplied groups to the CICS CSD:

```
//SYSIN  DD DSN=thlqual.SCSQPROC(CSQ4B100),DISP=SHR
//       DD DSN=thlqual.SCSQPROC(CSQ4D100),DISP=SHR
//       DD DSN=thlqual.SCSQPROC(CSQ4S100),DISP=SHR
//       DD *
  ADD GROUP(CSQCAT1) LIST(yourlist)
  ADD GROUP(CSQKDQ1) LIST(yourlist)
  ADD GROUP(CSQ4SAMP) LIST(yourlist)
/*
```

*Figure 9. JCL fragment for upgrading the CICS CSD*

Here, `yourlist` is the name of a CICS list that contains a list of groups to be installed by CICS during a cold start of the system. This is specified in the GRPLIST parameter of your CICS system initialization table (SIT). For details of CICS SIT parameters, see the *CICS System Definition Guide*.

Include the new resource groups in the CICS startup group list. For information about resource groups, installing them in CICS, the CICS CSD, and DFHCSDUP, see the *CICS Resource Definition Guide*.

**Note:** If you use the CEDA transaction to install redefined adapter resources in an active CICS system, you must first shut down the adapter and wait until the alert monitor has finished its work.

| If you want to use CICS program autoinstall rather than define the programs to
| the CICS CSD, you must ensure that the required programs are available to
| MQSeries. To do this, ensure that the autoinstalled definitions map to those
| supplied in member thlqual.SCSQPROC(CSQ4B100).

## Starting a connection automatically during CICS initialization

If you want the adapter to connect to MQSeries automatically during CICS initialization, the CSQCCODF program should be included in a CICS PLTPI program. CSQCCODF must execute during the third stage of CICS initialization and must therefore be added after the entry for DFHDELIM. If there is no entry for DFHDELIM in your current PLTPI, you must add one.

Alternatively, if your version of CICS supports it, you can use the MQCONN SIT parameter to connect to MQSeries automatically. See the *CICS System Definition Guide* for information about this parameter.

Instead of using CSQCCODF, you can write your own program; see “Writing a PLTPI program to start the connection” on page 89.

1. Use the CICS DFHPLT macro to add your program to the list of programs executed by CICS during the third stage initialization. Figure 10 shows how to code the entry for CSQCCODF in a CICS PLT program called DFHPLT41. For information about coding PLT entries, see the *CICS Resource Definition Guide*.

```
DFHPLT41 DFHPLT TYPE=INITIAL,SUFFIX=41
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
         DFHPLT TYPE=ENTRY,PROGRAM=CSQCCODF
         DFHPLT TYPE=FINAL
         END
```

*Figure 10. Sample PLT for use with the CICS adapter.* This sample assumes that you are using the supplied PLTPI program, CSQCCODF, to start the adapter.

2. Specify the particular list of programs to be run at initialization by naming the suffix of your PLT on the PLTPI system initialization parameter. In Figure 10, the PLT suffix is 41.

**Note:** You can use the CICS adapter in a CICS system that has interregion communication (IRC) to remote CICS systems. If you are using IRC, you should ensure that the IRC facility is OPEN before you start the adapter. This is essential if the IRC access method is defined as cross memory, that is, ACCESSMETHOD(XM).

## System definition

Use the INITPARM parameter in the CICS system initialization table (SIT), or the SYSIN override, to set the default connection parameters. Figure 11 shows you how to do this.

```
INITPARM=(CSQCPARM='SN=CSQ1,TN=001,IQ=CICS01.INITQ')
```

*Figure 11. Sample INITPARM statement to set the default connection values for CICS*

Where:

| | |
|---|---|
| **SN** | The subsystem name. This must be the name of a queue manager, not a queue-sharing group. |
| **TN** | The trace number to identify the adapter in CICS trace entries. This must be in the range 0 through 199. |
| **IQ** | The name of the default initiation queue. If this is blank, and you do not specify an initiation queue name by any other method, an instance of CKTI is not started when the CICS adapter connects to the queue manager. |

The INITPARM statement does not accept a parameter string longer than 60 characters. If you specify a 4-character subsystem name and a 3-character trace number, the maximum allowable length of the initiation queue name is 42 characters. If you need a queue name longer than 42 characters, you cannot use the INITPARM statement to specify the default initiation queue.

At connect time, you must override the INITPARM setting, either by using the CKQC transaction, or in a PLTPI program.

### Setting up the CICS adapter

1. If you are using a PLTPI program to start the adapter, code the suffix of your PLT on the PLTPI system initialization parameter. See Figure 10 on page 87 for an example of this.
2. Add the sample DCT entries (CSQ4DCT1 and CSQ4DCT2) to those of the existing CICS DCT and then reassemble the DCT.

   If you are not using the CICS mover, only the CKQQ definition in CSQ4DCT2 is needed.
3. Add the following MQSeries libraries to the STEPLIB concatenation in your CICS procedure in the following order, if they are not in the LPA or link list:
   - thlqual.SCSQANLx
   - thlqual.SCSQAUTH

   Where x is the language letter for your national language.
4. Add the following MQSeries libraries to the DFHRPL concatenation in your CICS procedure in the following order, if they are not in the LPA or link list:
   - thlqual.SCSQANLx
   - thlqual.SCSQCICS
   - thlqual.SCSQAUTH

   Where x is the language letter for your national language.

   If you are using any CICS programs that dynamically call the MQSeries CICS stub, CSQCSTUB, also add thlqual.SCSQLOAD to the DFHRPL concatenation.

   If you are using the API-crossing exit (CSQCAPX), also add the name of the library that contains the load module for the program.
5. Update CSQINP2. You can use the sample CSQ4INYG, but you might need to change the initiation queue name to match your system definition.

For more information about:

- The CICS initiation queue, see the *MQSeries for OS/390 Concepts and Planning Guide*.
- The CKQC transaction, see the *MQSeries for OS/390 System Administration Guide*
- PLTPI programs, see “Writing a PLTPI program to start the connection” on page 89.
- Coding CICS system initialization parameters, see the *CICS System Definition Guide*.

## SNAP dumps

If the CICS adapter detects an unexpected MQSeries subsystem error, it issues an OS/390 SNAP dump to DDname CSQSNAP and issues reason code MQRC_UNEXPECTED_ERROR to the application. If the CSQSNAP DD statement was not in the CICS startup JCL, no dump will be taken. If this happens, you could include the CSQSNAP DD statement in the startup JCL and rerun the application. However, because some problems might be intermittent, it is recommended that you include the CSQSNAP DD statement to capture the reason for failure at the time it occurs.

## Completing the connection from CICS

The connection is completed when the CICS adapter completes these steps:

1. Enable the CICS adapter and initialize the control blocks.
2. Attach the OS/390 subtasks and identify CICS generic *applId* (as specified in the CICS system initialization parameters as the connection ID) to MQSeries. This is described in the *CICS System Definition Guide*.

These two steps are done for you automatically if you use the INITPARM parameter or the CKQC transaction (this is described in the *MQSeries for OS/390 System Administration Guide*). You can also use a PLTPI program to do this; see "Writing a PLTPI program to start the connection".

When the connection is complete, a pending event called a *termination notification* is activated. This pending event remains active until MQSeries terminates in either an orderly or a forced way. When the pending event expires (or matures), it causes a FORCE shutdown request to be issued to the CICS adapter, and the pending event is canceled.

## Controlling CICS application connections

Every CICS transaction that issues calls to MQSeries is assigned a unique thread ID to service the requests and keep track of changes made to MQSeries resources. The thread ID is created the first time a transaction issues an MQSeries request, and accompanies all subsequent MQSeries requests made by that transaction.

While executing work under the CICS main task TCB, the CICS adapter queues MQSeries requests for processing by any of the eight subtask TCBs. These subtask TCBs are attached by the adapter when the connection to MQSeries is established.

## Customizing the CICS adapter

You can customize the CICS adapter by:

- Writing a user version of CSQCCODF that can be included in a CICS PLTPI program. See "Writing a PLTPI program to start the connection" for more information.
- Writing an API-crossing exit program. See "The API-crossing exit" on page 90 for more information.

### Writing a PLTPI program to start the connection

You can write your own PLTPI program, based on the supplied assembler sample thlqual.SCSQASMS(CSQCSPLT).

Although this sample is written in assembler, you can write your own program in any language supported by CICS. A typical use of PLTPI programs is for overriding the INITPARM settings if your CICS adapter initiation queue name is too long. (You cannot use more than 42 characters for an initiation queue name in an INITPARM statement.) If your PLTPI program gets its input parameters from a data set, you do not need an INITPARM statement.

Your PLTPI program must link to the adapter connect program, thlqual.SCSQCICS(CSQCQCON), and pass a parameter list that specifies the connection values to be used. The parameter list is described in the *MQSeries for OS/390 System Administration Guide*. Figure 12 on page 90 shows the LINK command that your PLTPI program must issue. In this example, the parameter list is named CONNPL. Because no terminals are available at this stage of CICS start up, you must use the COMMAREA option to pass the parameter list.

```
EXEC CICS LINK PROGRAM('CSQCQCON')
          COMMAREA(CONNPL) LENGTH(length of CONNPL)
```

*Figure 12. Linking to the adapter connect program, CSQCQCON, from a PLT program.* The COMMAREA option is used, because no terminals are currently available.

For more information about writing CICS PLTPI programs, see the *CICS Customization Guide*.

## The API-crossing exit

MQSeries provides an API-crossing exit for use with the CICS adapter; it runs in the CICS address space. You can use this exit to intercept MQI calls as they are being run, for monitoring, testing, maintenance, or security purposes.

| The sample API-crossing exit is supplied in source form only. For more
| information about writing API-crossing exit programs, see the *MQSeries Application*
| *Programming Guide*.

**Note:** Using the API-crossing exit degrades MQSeries performance. You should plan your use of it carefully.

### Defining the exit program

Before the API-crossing exit can be used, an exit program load module must be available when the CICS adapter connects to MQSeries. The exit program is a CICS program that must be named CSQCAPX and reside in a library in the DFHRPL concatenation. CSQCAPX must be defined in the CICS system definition file (CSD) and must be enabled.

When CSQCAPX is loaded a confirmation message is written to the CICS adapter control panel, CKQC, or the console. If it cannot be loaded, a diagnostic message is displayed, but otherwise the application program runs normally.

# Chapter 6. Customizing the CICS bridge

This chapter describes what you have to do to customize the MQSeries-CICS bridge. The bridge is described in the *MQSeries for OS/390 Concepts and Planning Guide*.

**Prerequisite APARs**

To run 3270 transactions, you must be using CICS Transaction Server for OS/390 Release 2 or later. Release 2 requires APAR PQ32659, Release 3 requires APAR PQ23961.

Before you can run the bridge you must ensure that your OS/390 system has both the CICS and MQSeries components in place.

## Setting up CICS

1. Run the resource definition utility DFHCSDUP, using the sample thlqual.SCSQPROC(CSQ4CKBC) as input, to define the bridge transactions and programs:

| | |
|---|---|
| **CKBR** | Bridge monitor transaction |
| **CSQCBCDI** | Data conversion exit |
| **CSQCBR00** | Bridge monitor program |
| **CKBP** | Bridge ProgramLink transaction |
| **CSQCBP00** | Bridge ProgramLink program |
| **CSQCBP10** | Bridge ProgramLink abend handler program |
| **CSQCBTX** | Bridge error messages |
| **CSQCBE00** | 3270 bridge exit for MQSeries (CICS Transaction Server, Version 1.2) |
| **CSQCBE30** | 3270 bridge exit for MQSeries (CICS Transaction Server, Version 1.3) |
| **CBR1 through CBR8** | 3270 bridge transaction definitions |

2. Add the load library to the DFHRPL concatenation of your CICS startup JCL.
3. Add the group, CSQCKB, to your startup group list.

**Notes:**

1. The bridge uses CICS temporary storage IDs with the prefix CKB. You should make sure these are not recoverable.
2. By default, your CICS DPL programs will be run under transaction code CKBP. The transaction to be run can be specified in the MQCIH CICS-bridge header in the message. For more information, see the *MQSeries Application Programming Reference* manual. You will need to change the TASKDATALOC attribute to 'BELOW' if you are going to run 24-bit programs, otherwise you will get a CICS abend AEZC.

## Customizing the CICS bridge

If you want to run your programs under different transaction codes you will need to install copies of the definition of CKBP, changing the transaction name to the ones of your choice. DPL bridge transactions must not be routed to a remote system.

3. If you are using CICS Transaction Server, Version 1.1, you need to install and use SupportPac™ MA1E, which supports DPL programs only.

## Setting up MQSeries

1. Define a local queue for the request messages.

   You can use the sample thlqual.SCSQPROC(CSQ4CKBM) to define a queue named SYSTEM.CICS.BRIDGE.QUEUE, or define your own. If you define your own, you must set the following attributes:

   **SHARE**
   So that both the monitor and the bridge tasks can read it.

   **MSGDLVSQ(FIFO)**
   So that messages are processed in FIFO sequence (not priority sequence).

   If recovery is required, set the following attributes:

   **DEFPSIST(YES)**
   Set messages as persistent on the queue by default.

   **HARDENBO**
   Set HARDENBO to ensure that messages are not re-processed erroneously after an emergency restart.

   If the request queue is defined with QSGDISP(SHARED), you must also define it with INDXTYPE(CORRELID).

2. Define one or more queues to hold the responses, as required. If your response queue is remote, you must define a transmission queue to hold the responses before they are forwarded to the response queue.
3. Ensure that the LE/370 libraries are included in the CICS library concatenation.

If the bridge is to be accessed remotely from MQSeries for OS/390, you need channel and transmission queue definitions, and a remote queue definition for the request queue. For more information about using remote queues see the *MQSeries Intercommunication* manual.

**Note:** The MQSeries queue defined to hold requests for the CICS bridge must not be used by any other application. Each CICS bridge monitor task started requires its own MQSeries queue to hold requests.

### Security

You might need to add RACF definitions, depending on the authentication option you choose to use. See “Security considerations for the CICS bridge” on page 209 for more information about this.

# Part 3. Customizing for IMS

# Chapter 7. Setting up the IMS adapter

This section tells you how to make the MQSeries-IMS adapter (referred to in this book as the IMS adapter) available to your IMS subsystem. If you are not familiar with tailoring an IMS subsystem, see the *IMS Customization Guide*.

To make the IMS adapter available to IMS applications, follow these steps:

1. Define MQSeries to IMS as an external subsystem using the IMS external subsystem attach facility (ESAF). See “Defining MQSeries to IMS” on page 96.
2. Include the MQSeries load library thlqual.SCSQAUTH in the JOBLIB or STEPLIB concatenation in the JCL for your IMS control region and for any dependent region that connects to MQSeries (if it is not in the LPA or link list). If your JOBLIB or STEPLIB is not authorized, also include it in the DFSESL concatenation after the library containing the IMS modules (usually IMS RESLIB).

   Also include thlqual.SCSQANLx (where x is the language letter).
3. Copy the MQSeries assembler program CSQQDEFV from thlqual.SCSQASMS to a user library.
4. The supplied program, CSQQDEFV, contains one subsystem name CSQ1 identified as default with an IMS language interface token (LIT) of MQM1. You can retain this name for testing and installation verification. For production subsystems, you can change the NAME=CSQ1 to your own subsystem name or use CSQ1. You can add further subsystem definitions as required. See “Defining the MQSeries subsystem to the IMS adapter” on page 99.
5. Assemble and link-edit the program to produce the CSQQDEFV load module. For the assembly, include the library thlqual.SCSQMACS in your SYSLIB concatenation; use the link-edit parameters `RENT,AMODE=31,RMODE=ANY`. This is shown in the sample JCL in thlqual.SCSQPROC(CSQ4DEFV).
6. Include the user library containing the module CSQQDEFV that you created in the JOBLIB or STEPLIB concatenation in the JCL for your IMS control region and for any dependent region that connects to MQSeries. If you do not do this, you will receive a user 3041 abend from IMS.
7. If the IMS adapter detects an unexpected MQSeries subsystem error, it issues an OS/390 SNAP dump to DDname CSQSNAP and issues reason code MQRC_UNEXPECTED_ERROR to the application. If the CSQSNAP DD statement was not in the IMS dependent region JCL, no dump will be taken. If this happens, you could include the CSQSNAP DD statement in the JCL and rerun the application. However, because some problems might be intermittent, it is recommended that you include the CSQSNAP DD statement to capture the reason for failure at the time it occurs.
8. If you want to use dynamic MQI calls (described in the *MQSeries Application Programming Guide*), build the dynamic stub, as shown in Figure 13 on page 96.
9. If you want to use the IMS trigger monitor, define the IMS trigger monitor application CSQQTRMN, and perform PSBGEN and ACBGEN. See “Setting up the IMS trigger monitor” on page 101.
10. If you are using RACF to protect resources in the OPERCMDS class, ensure that your MQSeries system has authority to issue the MODIFY command to any IMS system to which it might connect.

```
//DYNSTUB EXEC PGM=IEWL,PARM='RENT,REUS,MAP,XREF'
//SYSPRINT DD  SYSOUT=*
//ACSQMOD  DD DISP=SHR,DSN=thlqual.SCSQLOAD
//IMSLIB   DD  DISP=SHR,DSN=ims.reslib
//SYSLMOD  DD  DISP=SHR,DSN=private.load¹
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,1)
//SYSLIN   DD  *
  INCLUDE ACSQMOD(CSQQSTUB)
  INCLUDE IMSLIB (DFSLI000)
  ALIAS MQCONN,MQCONNX,MQDISC    MQI entry points
  ALIAS MQGET,MQPUT,MQPUT1       MQI entry points
  ALIAS MQOPEN,MQCLOSE           MQI entry points
  ALIAS MQBACK,MQCMIT            MQI entry points
  ALIAS CSQBBAK,CSQBCMT          MQI entry points
  ALIAS MQINQ,MQSET              MQI entry points
  ALIAS DFSPLI,PLITDLI           IMS entry points
  ALIAS DFSCOBOL,CBLTDLI         IMS entry points
  ALIAS DFSFOR,FORTDLI           IMS entry points
  ALIAS DFSASM,ASMTDLI           IMS entry points
  ALIAS DFSPASCL,PASTDLI         IMS entry points
  ALIAS DFHEI01,DFHEI1           IMS entry points
  ALIAS DFSAIBLI,AIBTDLI         IMS entry points
  ALIAS DFSESS,DSNWLI,DSNHLI     IMS entry points
  MODE AMODE(31),RMODE(ANY)      Note RMODE
  NAME CSQQDYNS(R)
/*

¹Specify the name of a library accessible to IMS applications that
wish to make dynamic calls to MQSeries.
```

*Figure 13. Sample JCL to link-edit the dynamic call stub.* This includes the IMS language interface module and the MQSeries IMS stub CSQQSTUB.

## Defining MQSeries to IMS

An MQSeries instance must be defined to the control region, and to each dependent region accessing that MQSeries subsystem. To do this, you must create a subsystem member (SSM) in the IMS.PROCLIB library, and identify the SSM to the applicable IMS regions.

### Placing the subsystem member entry in IMS.PROCLIB

Each SSM entry in IMS.PROCLIB defines a connection from an IMS region to a different subsystem.

To name an SSM member, concatenate the value (one to four alphanumeric characters) of the IMSID field of the IMS IMSCTRL macro with any name (one to four alphanumeric characters) defined by your site.

One SSM member can be shared by all of the IMS regions, or a specific member can be defined for each region. This member contains as many entries as there are connections to external subsystems. Each entry is an 80-character record.

## Positional parameters

The fields in this entry are:

```
SSN,LIT,ESMT,RTT,REO,CRC
```

where:

**SSN**
Specifies the MQSeries subsystem name. It is required, and must contain one through four characters. This name must be the name you specified in the subsystem name table (see "Updating the subsystem name table" on page 12).

**LIT**
Specifies the language interface token (LIT) supplied to IMS. This field is required, its value must match one in the CSQQDEFV module.

**ESMT**
Specifies the external subsystem module table (ESMT). This table specifies which attachment modules must be loaded by IMS. CSQQESMT is the required value for this field.

**RTT**
This option is not supported by MQSeries.

**REO**
Specifies the region error option (REO) to be used if an IMS application tries to reference a non-operational external subsystem or if resources are unavailable at create thread time. This field is optional and contains a single character, which can be:

**R** Passes a return code to the application, indicating that the request for MQSeries services failed.

**Q** Abends the application with an abend code U3051, backs out activity to the last commit point, does a PSTOP of the transaction, and requeues the input message. This option only applies when an IMS application tries to reference a non-operational external subsystem or if the resources are unavailable at create thread time.

MQSeries completion and reason codes are returned to the application if the MQSeries problem occurs while MQSeries is processing the request; that is, after the adapter has passed the request on to MQSeries.

**A** Abends the application with an abend code of U3047 and discards the input message. This option only applies when an IMS application tries to reference a non-operational external subsystem or if the resources are unavailable at create thread time.

MQSeries completion and reason codes are returned to the application if the MQSeries problem occurs while MQSeries is processing the request; that is, after the adapter has passed the request on to MQSeries.

**CRC**
This option can be specified but is not used by MQSeries.

## Setting up the IMS adapter

An example SSM entry is:

```
CSQ1,MQM1,CSQQESMT,,R,
```

where:

| | |
|---|---|
| **CSQ1** | The default subsystem name as supplied with MQSeries. You can change this to suit your installation. |
| **MQM1** | The default LIT as supplied in CSQQDEFV. |
| **CSQQESMT** | The external subsystem module name. You must use this value. |
| **R** | REO option. |

### Keyword parameters

MQSeries parameters can be specified in keyword format; to do this you must specify SST=DB2. Other parameters are as described in "Positional parameters" on page 97, and shown in the following example:

```
SST=DB2,SSN=SYS3,LIT=MQM3,ESMT=CSQQESMT
```

where:

| | |
|---|---|
| **SYS3** | The subsystem name |
| **MQM3** | The LIT as supplied in CSQQDEFV |
| **CSQQESMT** | The external subsystem module name |

## Specifying the SSM EXEC parameter

Specify the SSM EXEC parameter in the start up procedure of the IMS control region. This parameter specifies the one-character to four-character subsystem member name (SSM).

If you specify the SSM for the IMS control region, any dependent region running under the control region can attach to the MQSeries subsystem named in the IMS.PROCLIB member specified by the SSM parameter. The IMS.PROCLIB member name is the IMS ID (IMSID=*xxxx*) concatenated with the one to four characters specified in the SSM EXEC parameter. The IMS ID is the IMSID parameter of the IMSCTRL generation macro.

IMS lets you define as many external subsystem connections as are required. More than one connection can be defined for different MQSeries subsystems. All MQSeries connections must be within the same OS/390 system. For a dependent region, you can specify a dependent region SSM or use the one specified for the control region. You can specify different region error options (REOs) in the dependent region SSM member and the control region SSM member. Table 12 shows the different possibilities of SSM specifications.

*Table 12. SSM specifications options*

| SSM for control region | SSM for dependent region | Action | Comments |
|---|---|---|---|
| No | No | None | No external subsystem can be connected. |

*Table 12. SSM specifications options (continued)*

| SSM for control region | SSM for dependent region | Action | Comments |
|---|---|---|---|
| No | Yes | None | No external subsystem can be connected. |
| Yes | No | Use the control region SSM | Applications scheduled in the region can access external subsystems identified in the control region SSM. Exits and control blocks for each attachment are loaded into the control region and the dependent region address spaces. |
| Yes | Yes (empty) | No SSM is used for the dependent region | Applications scheduled in this region can access DL/I databases only. Exits and control blocks for each attachment are loaded into the control region address space. |
| Yes | Yes (not empty) | Check the dependent region SSM with the control region SSM | Applications scheduled in this region can access only external subsystems identified in both SSMs. Exits and control blocks for each attachment are loaded into the control region and the dependent region address spaces. |

There is no specific parameter to control the maximum number of SSM specification possibilities.

## Defining the MQSeries subsystem to the IMS adapter

The IMS adapter cannot access the IMS PROCLIB so the names of the MQSeries subsystems and their corresponding LITs must be defined in the subsystem definition table, CSQQDEFV. Use the supplied CSQQDEFX macro to create the CSQQDEFV load module. Figure 14 shows the syntax of this assembler macro.

```
 CSQQDEFX  TYPE=ENTRY|DEFAULT,NAME=subsystem,LIT=token
  or
 CSQQDEFX  TYPE=END
```

*Figure 14. CSQQDEFX macro syntax*

### Parameters

**TYPE=ENTRY|DEFAULT**
Specify either TYPE=ENTRY or TYPE=DEFAULT as follows:

**TYPE=ENTRY**
Specifies that a table entry describing an MQSeries subsystem available to an IMS application is to be generated. If this is the first entry, the table header is also generated, including a CSQQDEFV CSECT statement.

**TYPE=DEFAULT**
| As for TYPE=ENTRY. The subsystem specified is the default subsystem to
| be used when **MQCONN** or **MQCONNX** specifies a name that is all
| blanks. There must be only one such entry in the table.

**NAME=***subsystem*
| Specifies the name of the subsystem, as specified with **MQCONN** or
| **MQCONNX**.

**LIT=***token*
Specifies the name of the language interface token (LIT) that IMS uses to identify the subsystem. The LIT for each CSQQDEFX entry must be unique.

| An **MQCONN** or **MQCONNX** call associates the *name* input parameter and
| the *hconn* output parameter with the name label and, therefore, the LIT in the
| CSQQDEFV entry. Further MQSeries calls passing the *hconn* parameter use the
| LIT from the CSQQDEFV entry identified in the **MQCONN** or **MQCONNX**
| call to direct calls to the MQSeries instance defined in the IMS SSM PROCLIB
| member with that same LIT.

| In summary, the *name* parameter on the **MQCONN** or **MQCONNX** call
| identifies a LIT in CSQQDEFV and the same LIT in the SSM member identifies
| an MQSeries instance. (For information about the **MQCONN** and **MQCONNX**
| calls, see the *MQSeries Application Programming Reference* manual.)

**TYPE=END**
Specifies that the table is complete. If this parameter is omitted, TYPE=ENTRY is assumed.

## Using the CSQQDEFX macro

Figure 15 shows the general layout of a subsystem definition table.

```
CSQQDEFX NAME=subsystem1,LIT=token1
CSQQDEFX NAME=subsystem2,LIT=token2,TYPE=DEFAULT
CSQQDEFX NAME=subsystem3,LIT=token3
  ...
CSQQDEFX NAME=subsystemN,LIT=tokenN
CSQQDEFX TYPE=END
END
```

*Figure 15. Layout of a subsystem definition table*

## Setting up the IMS trigger monitor

Define the application to IMS using the model CSQQTAPL in the thlqual.SCSQPROC library (see Figure 16).

Generate the PSB and ACB using the model CSQQTPSB in the thlqual.SCSQPROC library (see Figure 17).

```
         TITLE 'CSQQTAPL - Transaction Definition for CSQQTRMN'
         SPACE 1
*                                                                  *
********************************************************************
*                                                                  *
*   This is the application definition                             *
*   for the IMS Trigger Monitor BMP                                *
*                                                                  *
*   The class parameter on the PGMTYPE keyword can be modified     *
*   to meet installation conventions.                              *
*                                                                  *
********************************************************************
         SPACE 1
         APPLCTN PSB=CSQQTRMN,                                     X
               PGMTYPE=BATCH,                                      X
               SCHDTYP=PARALLEL
         SPACE 1
```

*Figure 16. Example CSQQTAPL transaction definition for CSQQTRMN*

```
         TITLE 'CSQQTPSB - PSB for IMS trigger monitor'
         SPACE 1
********************************************************************
*                                                                  *
* This is the PSB for the MQSeries IMS trigger monitor program,    *
*                   CSQQTRMN.                                      *
*                                                                  *
********************************************************************
   SPACE 1
   PCB   TYPE=TP,           ALTPCB for transaction messages      X
         MODIFY=YES,        To "triggered" IMS transaction       X
         PCBNAME=CSQQTRMN
   PCB   TYPE=TP,           ALTPCB for diagnostic messages       X
         MODIFY=YES,        To LTERM specified or "MASTER"       X
         PCBNAME=CSQQTRMG,                                       X
         EXPRESS=YES
   PSBGEN LANG=ASSEM,                                            X
         PSBNAME=CSQQTRMN,   Runs program CSQQTRMN               X
         CMPAT=YES
   END
```

*Figure 17. Example CSQQTPSB PSB definition for CSQQTRMN*

# Chapter 8. Customizing the IMS bridge

This chapter describes what you have to do to customize the MQSeries-IMS bridge. The bridge is described in the *MQSeries for OS/390 Concepts and Planning Guide*.

**Define the XCF and OTMA parameters for MQSeries.**
This step defines the XCF group and member names for your MQSeries system, and other OTMA parameters. MQSeries and IMS must belong to the same XCF group. Use the OTMACON keyword of the CSQ6SYSP macro to tailor these parameters in the system parameter load module.

See "Using CSQ6SYSP" on page 31 for information about this.

**Define the XCF and OTMA parameters to IMS.**
This step defines the XCF group and member names for the IMS system. IMS and MQSeries must belong to the same XCF group.

Add the following parameters to your IMS parameter list, either in your JCL or in member DFSPBxxx in the IMS PROCLIB:

**OTMA=Y**
This starts OTMA automatically when IMS is started. (This is optional, if you specify OTMA=N you can also start OTMA by issuing the IMS command /START OTMA.)

**GRNAME=**
This gives the XCF group name.

This is the same as the group name specified in the storage class definition (see the next step), and in the Group parameter of the OTMACON keyword of the CSQ6SYSP macro.

**USERVAR=**
This gives the XCF member name of the IMS system.

This is the same as the member name specified in the storage class definition (see the next step).

If you do not specify a name for USERVAR, the value of APPLID1 is used.

**Tell MQSeries the XCF group and member name of the IMS system.**
This is specified by the storage class of a queue. If you want to send messages across the MQSeries-IMS bridge you need to specify this when you define the storage class for the queue. In the storage class, you need to define the XCF group and the member name of the target IMS system. To do this, either use the MQSeries operations and control panels, or use the MQSC commands as described in the *MQSeries MQSC Command Reference* manual.

**Set up the security that you require.**
| The /SECURE OTMA IMS command determines the level of security to be
| applied to ***every*** MQSeries subsystem that connects to IMS through OTMA.
| See "Security considerations for the IMS bridge" on page 211 for
| information about what this should be set to.

# Part 4. Monitoring performance and resource usage

# Chapter 9. Introduction to monitoring

This section describes how to monitor the performance and resource usage of an MQSeries subsystem.

- It outlines some of the information that you can retrieve and briefly describes a general approach to investigating performance problems. (You can find information about dealing with performance problems in the *MQSeries for OS/390 Problem Determination Guide*.)
- It describes how you can collect statistics about the performance of an MQSeries subsystem by using SMF records.
- It describes how to gather accounting data to enable you to charge your customers for their use of your MQSeries subsystems.
- It describes how to use MQSeries events (alerts) to monitor your systems.

These are some of the tools you might use to monitor MQSeries; they are described in the sections that follow:

- Tools provided by MQSeries:
  - "Using DISPLAY commands"
  - "Using CICS adapter statistics" on page 108
  - "Using MQSeries events" on page 110
- OS/390 service aids:
  - "Using System Management Facility" on page 110
- Other IBM licensed programs:
  - "Using Resource Measurement Facility" on page 112
  - "Using Performance Reporter for OS/390" on page 112
  - "Using the CICS monitoring facility" on page 112

Information about interpreting the data gathered by the performance statistics trace is given in "Chapter 10. Interpreting MQSeries performance statistics" on page 115.

Information about interpreting the data gathered by the accounting trace is given in "Chapter 11. Interpreting MQSeries accounting data" on page 127.

## Getting snapshots of MQSeries

You can get an idea of the current state of your MQSeries subsystem by using the DISPLAY commands and, for the CICS adapter, the CICS adapter panels.

### Using DISPLAY commands

You can use the MQSeries DISPLAY commands to obtain information about the current state of MQSeries. They provide information on the status of the command server, process definitions, queues, the queue manager, and so on. These commands are:

- DISPLAY CHANNEL
- DISPLAY CHSTATUS
- DISPLAY CMDSERV
- DISPLAY CLUSQMGR
- DISPLAY DQM
- | DISPLAY GROUP
- DISPLAY PROCESS

- DISPLAY QMGR
- DISPLAY QSTATUS
- DISPLAY QUEUE
- DISPLAY SECURITY
- DISPLAY STGCLASS
- DISPLAY THREAD
- DISPLAY TRACE
- DISPLAY USAGE

These commands provide a snapshot of the system *only* at the moment the command was processed. If you want to examine trends in the system, you must start an MQSeries trace and analyze the results over a period of time.

For the detailed syntax of each command, see the *MQSeries MQSC Command Reference* manual. All of the functions of these commands (except DISPLAY CMDSERV and DISPLAY TRACE) are also available through the operations and control panels.

## Using CICS adapter statistics

If you are an authorized CICS user, you can use the CICS adapter control panels to display CICS adapter statistics dynamically. These statistics provide a snapshot of information related to CICS thread usage and situations when all threads are busy. The display connection panel can be refreshed by pressing the Enter key. For more information, see the *MQSeries for OS/390 System Administration Guide*.

## Using MQSeries trace

You can record performance statistics and accounting data for MQSeries by using the MQSeries trace facility. The data generated by MQSeries is sent to:

- The System Management Facility (SMF), specifically as SMF record type 115, subtypes 1 and 2 for the performance statistics trace
- The SMF, specifically as SMF record type 116, subtypes 0, 1, and 2 for the accounting trace.

If you prefer, the data generated by the MQSeries accounting trace can also be sent to the generalized trace facility (GTF).

### Starting MQSeries trace

You can start the MQSeries trace facility at any time by issuing the MQSeries START TRACE command.

Accounting data can be lost if the accounting trace is started or stopped while applications are running. In order to collect accounting data successfully, the following conditions must apply:

- The accounting trace must be active when an application starts, and it must still be active when the application finishes.
- If the accounting trace is stopped, any accounting data collection that was active will cease.

You can also start collecting some trace information automatically if you specify YES on the SMFSTAT (SMF STATISTICS) and SMFACCT (SMF ACCOUNTING) parameters of the CSQ6SYSP macro (described in “Using CSQ6SYSP” on page 31).

| You cannot use this method to start collecting class 3 accounting information (thread-level and queue-level accounting). You must use the START TRACE command to do this (however, you can include the command in your CSQINP2 input data set so that the trace is started automatically when you start MQSeries).

Before starting an MQSeries trace, read “Using System Management Facility” on page 110.

## Controlling MQSeries trace

To control the MQSeries trace data collection at start up, specify values for the parameters in the CSQ6SYSP macro when you install MQSeries, see page 31.

You can control MQSeries tracing when MQSeries is running with these commands:
- START TRACE
- ALTER TRACE
- STOP TRACE

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

## Specifying trace keywords

The commands and keywords you can specify to control trace are described in the *MQSeries MQSC Command Reference* manual. When you specify a trace number, you must also specify the trace type.

### Specifying a destination

The DEST keyword specifies the location to which trace data is sent. Possible destinations are:

| | |
|---|---|
| **SMF** | System Management Facility |
| **GTF** | Generalized Trace Facility (accounting trace only) |
| **SRV** | Serviceability routine for diagnostic use by IBM service personnel |

For daily monitoring, information is sent to SMF (the default destination). SMF data sets usually contain information from other systems; this information is not available for reporting until the SMF data set is dumped.

| You can also send accounting trace information to the GTF. This information has an event identifier of 5EE. The *MQSeries for OS/390 Problem Determination Guide* describes how to deal with MQSeries trace information sent to the GTF.

## Effect of trace on MQSeries performance

Using the MQSeries trace facility can have a significant effect on MQSeries and transaction performance. For example, if you start a global trace for class 1 or for all classes, it is likely to increase CPU usage and transaction response times by approximately 50%. However, if you start a global trace for classes 2 to 4 alone, or a statistics or accounting trace, the increase in CPU usage and transaction response times is likely to be less than 1% additional CPU cost to the cost of MQSeries calls.

## Using MQSeries events

MQSeries *instrumentation events* provide information about errors, warnings, and other significant occurrences in a queue manager. You can monitor the operation of all your queue managers by incorporating these events into your own system management application.

MQSeries instrumentation events fall into the following categories:

**Queue manager events**
These events are related to the definitions of resources within queue managers. For example, an application attempts to put a message to a queue that does not exist.

**Performance events**
These events are notifications that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached, or the queue was not serviced within a predefined time limit.

**Channel events**
These events are reported by channels as a result of conditions detected during their operation. For example, when a channel instance is stopped.

**Note:** Channel events are not produced if you are using the “CICS mover” for distributed queuing.

When an event occurs, the queue manager puts an *event message* on the appropriate *event queue*, if defined. The event message contains information about the event that can be retrieved by a suitable MQI application.

MQSeries events can be enabled using the MQSC commands or the operations and control panels. Channel events can only be disabled by altering the definition of the event queue to PUT(DISABLED).

| See the *MQSeries Event Monitoring* manual for information about the MQSeries
| events that cause event messages to be generated, and for information about the
| format of these messages. See the *MQSeries MQSC Command Reference* for
| information about enabling the events.

## Using System Management Facility

System management facility (SMF) is an OS/390 service aid used to collect information from various OS/390 subsystems. This information is dumped and reported periodically, for example, hourly. You can use SMF with the MQSeries trace facility to collect data from MQSeries. In this way you can monitor *trends*, for example, in system utilization and performance, and collect accounting information about each user ID using the MQSeries subsystem.

To record performance statistics (record type 115) to SMF specify the following in the SMFPRMxx member of SYS1.PARMLIB or with the SETSMF OS/390 operator command.

```
SYS(TYPE(115))
```

To record accounting information (record type 116) to SMF specify the following in the SMFPRMxx member of SYS1.PARMLIB or with the SETSMF OS/390 operator command.

```
SYS(TYPE(116))
```

To use the SETSMF OS/390 operator command, either PROMPT(ALL) or PROMPT(LIST) must be specified in the SMFPRM*xx* member. See the *OS/390 MVS Initialization and Tuning Reference* and the *OS/390 MVS System Commands* manuals for more information.

You must also set the SMFSTAT and SMFACCT parameters to YES; this is described in “Using CSQ6SYSP” on page 31.

| You can specify the interval at which MQSeries collects statistics and accounting
| data in one of two ways:

- | You can specify a value for STATIME in your system parameters (described in
  | “Using CSQ6SYSP” on page 31).
- | You can specify 0 for STATIME and use the SMF global accounting interval
  | (described in the *OS/390 MVS Initialization and Tuning Reference*).

| SMF must be running before you can send data to it. For more information about SMF, see the *MVS System Management Facilities (SMF)* manual.

## Allocating additional SMF buffers

When you invoke a trace, you must ensure that you allocate adequate SMF buffers. Specify SMF buffering on the VSAM BUFSP parameter of the access method services DEFINE CLUSTER statement. Specify CISZ(4096) and BUFSP(81920) on the DEFINE CLUSTER statement for each SMF VSAM data set.

If an SMF buffer shortage occurs, SMF rejects any trace records sent to it. MQSeries sends a CSQW133I message to the OS/390 operator when this occurs. MQSeries treats the error as temporary and remains active even though SMF data could be lost. When the shortage has been alleviated and trace recording has resumed, MQSeries sends a CSQW123I message to the OS/390 operator.

## Reporting data in SMF

You can use the SMF program IFASMFDP to dump SMF records to a sequential data set so that they can be processed.

There are several ways to report on this data, for example:

- Write an application program to read and report information from the SMF data set. You can then tailor the report to fit your exact needs.
- | Use Performance Reporter to process the records (see “Using Performance
  | Reporter for OS/390” on page 112).

## Using other products with MQSeries

You can use other products to help you to improve the presentation of, or to augment statistics related to, performance and accounting.

### Using Resource Measurement Facility

Resource Management Facility (RMF) is an IBM licensed program (program number 5685-029) that provides system-wide information on processor utilization, I/O activity, storage, and paging. You can use RMF™ to monitor the utilization of physical resources across the whole system dynamically. For more information, see the *MVS Resource Measurement Facility User's Guide*.

### Using Performance Reporter for OS/390

You can use Performance Reporter for OS/390 to interpret RMF and SMF records.

Performance Reporter for OS/390 is an IBM licensed program (program number 5695-101) that enables you to manage the performance of your system by collecting performance data in a DB2 database and presenting the data in a variety of formats for use in systems management. Performance Reporter can generate graphic and tabular reports using systems management data it stores in its DB2 database. It includes an administration dialog, a reporting dialog, and a log collector, all of which interact with a standard DB2 database.

This is described in the *Performance Reporter for OS/390 Administration Guide*.

### Using the CICS monitoring facility

The CICS monitoring facility provides performance information about each CICS transaction running. It can be used to investigate the resources used and the time spent processing transactions. For background information, see the *CICS Performance Guide* and the *CICS Customization Guide*.

## Investigating performance problems

Performance can be adversely affected by:

- Buffer pools that are an incorrect size
- Lack of real storage
- I/O contention for page sets or logs
- Log buffer thresholds that are set incorrectly
- Incorrect setting of the number of log buffers
- Large messages
- Units of recovery that last a long time, incorporating many messages per syncpoint
- Messages that remain on a queue for a long time
- RACF auditing
- Unnecessary security checks
- Inefficient program design

When you analyze performance data, always start by looking at the overall system before you decide that you have a specific MQSeries problem. Remember that almost all symptoms of reduced performance are magnified when there is contention. For example, if there is contention for DASD, transaction response times can increase. Also, the more transactions there are in the system, the greater the processor overhead and greater the demand for both virtual and real storage.

In such situations, the system shows heavy use of *all* its resources. However, the system is actually experiencing normal system stress, and this might be hiding the cause of a performance reduction. To find the cause of such a loss of performance, you must consider all items that might be affecting your active tasks.

### Investigating the overall system

Within MQSeries, the performance problem is either reduced response time or an unexpected and unexplained heavy use of resources. You should first check factors such as total processor usage, DASD activity, and paging. An IBM tool for this is resource management facility (RMF). In general, you need to look at the system in some detail to see why tasks are progressing slowly, or why a given resource is being heavily used.

Start by looking at general task activity, then focus on particular activities, such as specific tasks or a specific time interval.

Another possibility is that the system has limited real storage; therefore, because of paging interrupts, the tasks progress more slowly than expected.

| 
### Investigating individual tasks

| You can use the accounting trace to gather information about MQSeries tasks.
| These trace records tell you a great deal about the activity that the task has
| performed, and about how much time the task spent suspended, waiting for
| latches. The trace record also includes information about how much DB2 and
| Coupling Facility activity was performed by the task.

| This is described in “Chapter 11. Interpreting MQSeries accounting data” on
| page 127.

# Chapter 10. Interpreting MQSeries performance statistics

MQSeries performance statistics are written as SMF type 115 records. Statistics records are produced periodically at a time interval specified by the STATIME parameter of the CSQ6SYSP system parameter module, or at the SMF global accounting interval if you specify 0 for STATIME. The information provided in the SMF records comes from the following components of MQSeries:

| | |
|---|---|
| **Buffer manager** | Manages the buffer pools in virtual storage and the writing of pages to page sets as the buffer pools become full. Also manages the reading of pages from page sets. |
| **Coupling Facility manager** | Manages the interface with the Coupling Facility. |
| **Data manager** | Manages the links between messages and queues. It calls the buffer manager to process the pages with messages on them. |
| **DB2 manager** | Manages the interface with the DB2 database that is used as the shared repository. |
| **Lock manager** | Manages locks for MQSeries for OS/390. |
| **Log manager** | Manages the writing of log records, which are essential for maintaining the integrity of the system if there is a back out request, or for recovery, if there is a system or media failure. |
| **Message manager** | Processes all MQI requests. |
| **Storage manager** | Manages storage for MQSeries for OS/390, for example, storage pool allocation, expansion, and deallocation. |

MQSeries statistics can be collected for two subtypes:

**1** System information, for example, related to the logs and storage.

**2** Information about number of messages, buffer and paging information. Queue-sharing group information related to the Coupling Facility and DB2.

The subtype is specified in the SM115STF field (shown in Table 13 on page 116).

## Layout of an SMF type 115 record

The standard layout for SMF records involves three parts:

**SMF header**
Provides format, identification, and time and date information about the record itself.

**Self-defining section**
Defines the location and size of the individual data records within the SMF record.

**Data records**
The actual data from MQSeries that you want to analyze.

For more information about SMF record formats, see the *MVS System Management Facilities (SMF)* manual.

## The SMF header

Table 13 shows the format of SMF record header (SM115).

*Table 13. SMF record header description*

| Offsets | | Type | Len | Name | Description | Example |
|---|---|---|---|---|---|---|
| **Dec** | **Hex** | | | | | |
| 0 | 0 | Structure | 28 | SM115 | SMF record header. | |
| | | Unsigned | 2 | SM115LEN | SMF record length. | 14A0 |
| 2 | 2 | | 2 | | Reserved. | 0000 |
| 4 | 4 | Unsigned | 1 | SM115FLG | System indicator. | 5E |
| 5 | 5 | Unsigned | 1 | SM115RTY | Record type. The SMF record type, for MQSeries statistics records this is always 115 (X'73'). | 73 |
| 6 | 6 | Unsigned | 4 | SM115TME | Time when SMF moved record. | 00355575 |
| 10 | A | Unsigned | 4 | SM115DTE | Date when SMF moved record. | 0100223F |
| 14 | E | Character | 4 | SM115SID | OS/390 subsystem ID. Defines the OS/390 subsystem on which the records were collected. | D4E5F4F1 (MV41) |
| 18 | 12 | Character | 4 | SM115SSI | MQSeries subsystem ID. | D4D8F0F7 (MQ07) |
| 22 | 16 | Unsigned | 2 | SM115STF | Record subtype. | 0002 |
| 24 | 18 | | 4 | | Reserved. | 00000000 |
| 28 | 1C | Character | 0 | SM115END | End of SMF header and start of self-defining section. | |
| **Note:** The (hexadecimal) values in the right-hand column relate to Figure 18. | | | | | | |

## Self-defining sections

A self-defining section of a type 115 SMF record tells you where to find a statistics record, how long it is, and how many times that type of record is repeated (with different values). The self-defining sections follow after the header, at fixed offsets from the start of the SMF record. Each statistics record can be identified by an eye-catcher string.

Eight types of self-defining section are available to users for type 115 records. Each self-defining section points to statistics data related to one of eight MQSeries components. Table 14 summarizes the sources of the statistics, the eye-catcher strings, and the offsets of the self-defining sections from the start of the SMF record header.

*Table 14. Offsets to self-defining sections.* Offsets are from the start of the SMF record and are fixed for each type of statistics source.

| Source of statistics | Record subtype (SM115STF) | Offset | Length of data | Eyecatcher |
|---|---|---|---|---|
| Storage manager | 1 | (X'64') | (X'48') | QSST |
| Log manager | 1 | (X'74') | (X'78') | QJST |
| Message manager | 2 | (X'24') | (X'30') | QMST |
| Data manager | 2 | (X'2C') | (X'50') | QIST |

*Table 14. Offsets to self-defining sections (continued).* Offsets are from the start of the SMF record and are fixed for each type of statistics source.

| Source of statistics | Record subtype (SM115STF) | Offset | Length of data | Eyecatcher |
|---|---|---|---|---|
| Buffer manager - one per buffer pool (4) | 2 | (X'34') | (X'68') | QPST |
| Lock manager | 2 | (X'3C') | (X'20') | QLST |
| DB2 manager | 2 | (X'44') | (X'1E0') | Q5ST |
| Coupling Facility manager | 2 | (X'4C') | (X'1008') | QEST |
| **Note:** Other self-defining sections refer to data for IBM use only. | | | | |

Each self-defining record is two fullwords long and has this format:

```
    ssssssssllllnnnn
```

where:

| | |
|---|---|
| **ssssssss** | Fullword containing the offset from the start of the SMF record. |
| **llll** | Halfword giving the length of this data record. |
| **nnnn** | Halfword giving the number of data records in this SMF record. |

For example, in Figure 18 on page 118, the self-defining section for message manager statistics is shown in bold. It is located at offset X'20' from the start of the SMF record and contains this information:

- The offset of the message manager statistics is located X'00000054' bytes from the start of the SMF record.
- The message manager record is X'0030' bytes long.
- There is one record (X'0001').

Similarly, in Figure 18, the buffer manager self-defining section at X'30' specifies that the offset to the buffer manager statistics is X'000000D4', is of length X'0068', and occurs X'0004' times.

**Note:** Always use offsets in the self-defining sections to locate the statistics records.

## Example SMF header and self-defining section

Figure 18 shows an example of part of an SMF type 115 subtype 2 record. The numbers in the left-hand column represent the offset, in hexadecimal, from the start of the record. Each line corresponds to sixteen bytes of data, where each byte is two hexadecimal characters, for example 0C. The characters in the right-hand column represent the printable characters for each byte. Non-printable characters are shown by a period (.) character.

In this example, alternate fields in the SMF header are underlined to help you to see them; refer to Table 13 to identify them. The self defining section for the message manager statistics data records (at the offset given in Table 14 on page 116) is shown in bold.

```
000000  14A00000 5E730035 55750100 223FD4E5 *....;.........MV*
000010  F4F1D4D8 F0F70002 00000000 0000147C *41MQ07.........@*
000020  00240001 00000054 00300001 00000084 *................*
000030  00500001 000000D4 00680004 00000274 *.&.....M........*
000040  00200001 00000294 01E00001 00000474 *................*
000050  10080001 D40F0030 D8D4E2E3 00000000 *....M...QMST....*
```

*Figure 18. Part of an SMF record showing the header and self-defining sections*

## Examples of SMF statistics records

Figure 19 shows an example of part of the SMF record for subtype 1. Subtype 1 includes the storage manager and log manager statistics records. The SMF record header and self-defining sections are shown underlined.

The self-defining section at offset X'64' refers to storage manager statistics and the self-defining section at offset X'74' refers to log manager statistics.

The storage manager statistics record is located at offset X'0000011C' from the start of the header and is X'48' bytes long. There is one set of storage manager statistics, identified by the eye-catcher string QSST. The start of this statistics record is also shown in the example.

The log manager statistics record is located at offset X'00000164' from the start of the header and is X'78' bytes long. There is one set of log manager statistics, identified by the eye-catcher string QJST.

```
000000  02000000 5E730035 55750100 223FD4E5 *....;.........MV*
000010  F4F1D4D8 F0F70001 00000000 000001DC *41MQ07..........*
000020  00240001 00000000 00000000 00000000 *................*
000030  00000000 00000000 00000000 0000007C *...............@*
000040  00400001 000000BC 00600001 00000000 *. .......-......*
000050  00000000 00000000 00000000 00000000 *................*
000060  00000000 0000011C 00480001 00000000 *................*
000070  00000000 00000164 00780001 00000000 *................*
000080  00000000 00000000 00000000 00000000 *................*
.
.
000110  00000000 00000000 00000000 003C0048 *................*
000120  D8E2E2E3 0000004F 00000003 00000002 *QSST...|........*
```

*Figure 19. SMF record 115, subtype 1*

Figure 20 shows an example of part of the SMF record for subtype 2. Subtype 2 includes the statistics records for the message, data, buffer, lock, Coupling Facility, and DB2 managers. The SMF record header and alternate self-defining sections are shown underlined.

- The self-defining section at offset X'24' refers to message manager statistics. The message manager statistics record is located at offset X'00000054' from the start of the header and is X'30' bytes long. There is one set of these statistics, identified by the eye-catcher string QMST.
- The self-defining section at offset X'2C' refers to data manager statistics. The data manager statistics record is located at offset X'00000084' from the start of the header and is X'50' bytes long. There is one set of these statistics, identified by the eye-catcher string QIST.
- The self-defining section at offset X'34' refers to buffer manager statistics. The buffer manager statistics record is located at offset X'000000D4' from the start of

the header and is X'68' bytes long. There are four sets of these statistics, identified by the eye-catcher string QPST.

- The self-defining section at offset X'3C' refers to lock manager statistics. The lock manager statistics record is located at offset X'00000274' from the start of the header and is X'20' bytes long. There is one set of these statistics, identified by the eye-catcher string QLST.
- The self-defining section at offset X'44' refers to DB2 manager statistics. The DB2 manager statistics record is located at offset X'00000294' from the start of the header and is X'1E0' bytes long. There is one set of these statistics, identified by the eye-catcher string Q5ST.
- The self-defining section at offset X'4C' refers to Coupling Facility manager statistics. The Coupling Facility manager statistics record is located at offset X'00000474' from the start of the header and is X'1008' bytes long. There is one set of these statistics, identified by the eye-catcher string QEST.

```
000000  14A00000 5E730035 55750100 223FD4E5 *....;.........MV*
000010  F4F1D4D8 F0F70002 00000000 0000147C *41MQ07.........@*
000020  00240001 00000054 00300001 00000084 *................*
000030  00500001 000000D4 00680004 00000274 *.&.....M........*
000040  00200001 00000294 01E00001 00000474 *................*
000050  10080001 D40F0030 D8D4E2E3 00000000 *....M...QMST....*
000060  00000000 00000000 00000000 00000000 *................*
000070  00000000 00000000 00000000 00000000 *................*
000080  00000000 C90F0050 D8C9E2E3 00000000 *....I..&QIST....*
000090  00000001 00000000 00000025 00000003 *................*
0000A0  00000000 0000002C 00000007 00000000 *................*
0000B0  00000000 00000000 00000000 00000012 *................*
0000C0  00000000 00000000 00000000 00000000 *................*
0000D0  00000000 D70F0068 D8D7E2E3 00000000 *....P...QPST....*
0000E0  000007D0 000007BD 000007BD 00000037 *...}............*
0000F0  00000000 0000001B 0000003B 00000000 *................*
000100  00000000 00000000 00000000 00000000 *................*
000110  0000001B 00000000 00000000 00000000 *................*
000120  00000000 00000000 00000000 00000000 *................*
000130  00000000 00000000 00000000 D70F0068 *............P...*
000140  D8D7E2E3 00000001 000007D0 000007CD *QPST.......}....*
.
.
```

*Figure 20. SMF record 115, subtype 2*

## Processing type 115 SMF records

You must process any data you collect from SMF to extract useful information. When you process the data, verify that the records are from MQSeries and that they are the records you are expecting.

Validate the values of the following fields:
- SM115RTY, the SMF record number, must be X'73' (115)
- SM115STF, the record subtype, must be 0001 or 0002

A program that you can use to format and print MQSeries statistics records is supplied in an MQSeries SupportPac.

## Storage manager data records

| The format of the storage manager statistics record is described in assembler macro
| thlqual.SCSQMACS(CSQDQSST).

### Interpretation

The data contains information about the number of fixed and variable storage pools that the queue manager has allocated, expanded, contracted and deleted during the statistics interval, plus the number of GETMAIN and FREEMAIN requests, including a count of those that were unsuccessful. Additional information includes a count of the number of times the SOS (short on storage) condition was detecting and a count of the number of abends that occurred as a result of the SOS condition.

## Log manager data records

| The format of the log manager statistics record is described in assembler macro
| thlqual.SCSQMACS(CSQDQJST).

### Interpretation

These counts are important:

1. The total number of log write requests:

   $N_{logwrite}$ = QJSTWRW + QJSTWRNW + QJSTWRF

2. The total number of log read requests:

   | $N_{logread}$ = QJSTRBUF + QJSTRACT + QJSTRARH

The problem symptoms that can be examined using log manager statistics are described in Table 15.

*Table 15. Problem symptoms that can be examined using log manager statistics*

| |
|---|
| **Symptom 1**<br>QJSTWTB is non-zero.<br>**Reason**<br>Tasks are being suspended while the in-storage buffer is being written to the active log.<br><br>There might be problems writing to the active log.<br><br>The OUTBUFF parameter within CSQ6LOGP is too small.<br>**Action**<br>Investigate the problems writing to the active log.<br><br>Increase the value of the OUTBUFF parameter within CSQ6LOGP. |
| **Symptom 2**<br>The ratio: QJSTWTL/$N_{logread}$ is greater than 1%.<br>**Reason**<br>Log reads were initiated that had to read from an archive log, but MQSeries was not able to allocate a data set because MAXALLC data sets were already allocated.<br>**Action**<br>Increase MAXALLC. |

*Table 15. Problem symptoms that can be examined using log manager statistics (continued)*

| |
|---|
| **Symptom 3**<br>The ratio: QJSTRARH/$N_{logread}$ is larger than normal.<br>**Reason**<br>Most log read requests should come from the output buffer or the active log. To satisfy requests for back out, unit-of-recovery records are read from the in-storage buffer, the active log, and the archived logs.<br><br>A long-running unit of recovery, extending over a period of many minutes, might have log records spread across many different logs. This degrades performance because extra work has to be done to recover the log records.<br>**Action**<br>Change the application to reduce the length of a unit of recovery. Also, consider increasing the size of the active log to reduce the possibility of a single unit of recovery being spread out over more than one log.<br>**Other pointers**<br>The ratio $N_{logread}$/$N_{logwrite}$ gives an indication of how much work has to be backed out. |
| **Symptom 4**<br>QJSTLLCP is more than 10 per hour.<br>**Reason**<br>On a busy system you would expect to see typically 10 checkpoints an hour. If the QJSTLLCP value is larger than this, it indicates a problem in the setup of the queue manager.<br><br>The most likely reason for this is that the LOGLOAD parameter in CSQ6SYSP is too small. The other event that causes a checkpoint is when an active log fills up and switches to the next active log data set. If your logs are too small, this can cause frequent checkpoints.<br>**Action**<br>Increase the LOGLOAD parameter, or increase the size of your log data sets as required. |

**Note:** In the first set of statistics produced after system startup, there might be significant log activity due to the resolution of in-flight units of recovery.

## Message manager data records

| The format of the message manager statistics record is described in assembler
| macro thlqual.SCSQMACS(CSQDQMST).

### Interpretation

The data gives you counts of different MQI requests.

## Data manager data records

| The format of the data manager statistics record is described in assembler macro
| thlqual.SCSQMACS(CSQDQIST).

### Interpretation

The data gives you counts of different object requests.

## Buffer manager data records

| The format of the buffer manager statistics record is described in assembler macro
| thlqual.SCSQMACS(CSQDQPST).

**Note:** If you have defined a buffer pool, but not used it, no values are set so the buffer manager statistics record will not contain any data.

### Interpreting buffer manager statistics

MQSeries writes statistics to SMF for each buffer pool, if statistics trace gathering has been requested. The statistics are reset each time they are output.

When interpreting the statistics, you are recommended to consider the following factors because the values of these fields can be used to improve the performance of your system:

1. If QPSTSOS, QPSTSTLA, or QPSTDMC is greater than zero, you should either increase the size of the buffer pool or reallocate the page sets to different buffer pools.
   - QPSTSOS is the number of times that there were no buffers available for page get requests. If QPSTSOS ever becomes non-zero, it shows that MQSeries is under severe stress. The buffer pool size should be significantly increased. If increasing the buffer pool size does not make the value of QPSTSOS zero, there might be I/O contention on the DASD page sets.
   - QPSTDMC is the number of updates that were performed synchronously because there was either more than 95% of the pages in the buffer pool waiting for write I/O, or there was less than 5% of the buffer pool available for read requests. If this number is not zero, the buffer pool might be too small and should be enlarged. If increasing the buffer pool size does not reduce QPSTDMC to zero, there might be I/O contention on the DASD page sets.
   - QPSTIMW is a count of the number of times pages were written out synchronously. If QPSTDMC is zero, QPSTIMW is the number of times pages were found on the queue waiting for write I/O that had been there for at least two checkpoints.
2. For buffer pool zero and buffer pools that contain short-lived messages:
   - QPSTDWT should be zero, and the percentage QPSTCBSL/QPSTNBUF should be greater than 15%.

     QPSTDWT is the number of times the asynchronous write processor was started because there was either more than 85% of the pages in the buffer pool waiting for write I/O, or there was less than 15% of the buffer pool available for read requests. Increasing the buffer pool size should reduce this value. If it does not, the pattern of access is one of long delays between puts and gets.
   - QPSTTPW might be greater than zero due to checkpointing activity.

- QPSTRIO should be zero unless messages are being read from a page set after the queue manager is restarted.

  The ratio of QPSTRIO to QPSTGETP shows the efficiency of page retrieval within the buffer pool. Increasing the buffer pool size should decrease this ratio and, therefore, increase the page retrieval efficiency. If this does not happen, it indicates that pages are not being frequently reaccessed. This implies a transaction pattern where there is a long delay between messages being put and then subsequently retrieved.

  The ratio of QPSTGETN to QPSTGETP indicates the number of times an empty page, as opposed to a non-empty page, has been requested. This ratio is more an indication of transaction pattern, than a value that can be used to tune the system.
- If QPSTSTL has a value greater than zero, this indicates that pages that have not been used before are now being used. This might be caused by an increased message rate, messages not being processed as fast as they were previously (leading to a buildup of messages), or larger messages being used.

  QPSTSTL is a count of the number of times a page access request did not find the page already in the buffer pool. Again, the lower the ratio of QPSTSTL to (QPSTGETP + QPSTGETN) is, the higher the page retrieval efficiency. Increasing the buffer pool size should decrease this ratio but, if it does not, it is an indication that there are long delays between puts and gets.
- You are recommended to have sufficient buffers to handle your peak message rate.

3. For buffer pools with long-lived messages, where there are more messages than will fit into the buffer pool:
   - (QPSTRIO+QPSTWIO)/Statistics interval is the I/O rate to page sets. If this value is high, you should consider using multiple page sets on different volumes to allow I/O to be carried out in parallel.

     The higher the ratio of QPSTSTW to QPSTWIO, the better the efficiency of the asynchronous write processor. You can increase this ratio, and therefore the efficiency of the asynchronous write processor, by increasing the buffer pool size.
   - Over the period of time that the messages are processed (for example, if messages are written to a queue during the day and processed overnight) the number of read I/Os (QPSTRIO) should be approximately the total number of pages written (QPSTTPW). This shows that one page is read for every page written.

     If QPSTRIO is much larger than QPSTTPW, this shows that pages are being read in multiple times. This might be a result of the application using **MQGET** by *MsgId* or *CorrelId* when the queue is not indexed, or browsing messages on the queue using get next.

     The following actions might relieve this problem:

     a. Increase the size of the buffer pool so that there are enough pages to hold the queue, in addition to any changed pages.
     b. Use the INDXTYPE queue attribute, which allows a queue to be indexed by *MsgId* or *CorrelId* and eliminates the need for a sequential scan of the queue.
     c. Change the design of the application to eliminate the use of **MQGET** with *MsgId* or *CorrelId*, or the get next with browse option.

**Note:** Applications using long-lived messages typically process the first available message and do not use **MQGET** with *MsgId* or *CorrelId*, and they might browse only the first available message.

d. Move page sets to a different buffer pool to reduce contention between messages from different applications.

## Managing your buffer pools

To manage your buffer pools efficiently, you must consider the factors that affect the buffer pool I/O operations and also the statistics associated with the buffer pools.

The following factors affect buffer pool I/O operations.

- If a page containing the required data is not found in the buffer pool, it is read in synchronously to an available buffer from its DASD page set.
- Whenever a page is updated, it is put on an internal queue of pages to be (potentially) written out to DASD. This means that the buffer used by that page is unavailable for use by any other page until the buffer has been written to DASD.
- If the number of pages queued to be written to DASD exceeds 85% of the total number of buffers in the pool, an asynchronous write processor is started in order to put the buffers to DASD.

  Similarly, should the number of buffers available for page get requests become less than 15% of the total number of buffers in the pool, the asynchronous write processor is started in order to perform the write I/O operations.

  The write processor stops when the number of pages queued to be written to DASD has fallen to 75% of the total number of buffer in the pool.
- If the number of pages queued for writing to DASD exceeds 95% of the total number of buffers in the pool, all updates result in a synchronous write of the page to DASD.

  Similarly, if the number of buffers available for page get requests becomes less than 5% of the total number of buffers in the pool, all updates result in a synchronous write of the page to DASD.
- If the number of buffers available for page get requests ever reaches zero, a transaction that encounters this condition is suspended until the asynchronous write processor has finished.
- If a page is frequently updated, the page spends most of its time on the queue of pages waiting to be written to DASD. Because this queue is in least recently used order, it is possible that a frequently updated page placed on this least recently used queue will never be written out to DASD. For this reason, at the time of update, if the page is found to have been waiting on the write to DASD queue for at least 2 checkpoints, it will be synchronously written to DASD. Updating occurs at checkpoint time.

  The aim of this algorithm is to maximize the time pages spend in buffer pool memory while allowing the system to function if the system load puts the buffer pool usage under stress.

## Lock manager data records

The format of the lock manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQLST).

### Interpretation

The data contains information about the following:

- The number of lock get requests and lock release requests.
- The number of times a lock get request determined that the requested lock was already held.

## DB2 manager data records

The format of the DB2 manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQ5ST). If the queue manager was not started as a member of a queue-sharing group, no data is recorded in this record.

### Interpretation

The data contains counts for each of the request types that the DB2 resource manager supports. For these request types, maximum and cumulative elapse times are kept for the following:

- The time spent in the DB2 resource manager as a whole (called the thread time).
- The time that was spent performing the RRSAF and SQL parts of the request (a subset of the thread time called the SQL time).

Information is also provided for:

- The number of server tasks attached.
- The maximum overall request depth against any of the server tasks.
- The number of times any of the server task requests terminated abnormally.

If the abnormal termination count is not zero, a requeue count is provided indicating the number of queued requests that were requeued to other server tasks as a result of the abnormal termination.

If the average thread time is significantly greater that the average SQL time, this might indicate that thread requests are spending an excessive amount of time waiting for a server task to process the SQL part of the request. If this is the case, examine the DHIGMAX field and, if the value is greater than one, consider increasing the number of DB2 server tasks specified in the QSGDATA parameter of the CSQ6SYSP system parameter macro.

## Coupling Facility manager data records

The format of the Coupling Facility manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQEST). If the queue manager was not started as a member of a queue-sharing group, no data is recorded in this record.

### Interpretation

The data contains information for each of the Coupling Facility list structures, including the CSQ_ADMIN structure, that the queue manager could connect to during the statistics interval. The information for each structure includes the following:

- The number of and cumulative elapse times for IXLLSTE and IXLLSTM requests.
- The number of times a request had to be retried because of a timeout.
- The number of times a ‘structure full’ condition occurred.

# Chapter 11. Interpreting MQSeries accounting data

MQSeries accounting data is written as SMF type 116 records.

MQSeries accounting information can be collected for three subtypes:

**0** Message manager accounting records (how much CPU was spent processing MQI calls and the number of **MQPUT** and **MQGET** calls). This information is produced when a named task disconnects from MQSeries, and so the information contained within the record might cover many hours.

**1** Accounting data for each task, at thread and queue level.

**2** Additional queue-level accounting data (if the task used more queues than could fit in the subtype 1 record).

## Layout of an SMF type 116 record

The standard layout for SMF records involves three parts:

**SMF header**
Provides format, identification, and time and date information about the record itself.

**Self-defining section**
Defines the location and size of the individual data records within the SMF record.

**Data records**
The actual data from MQSeries that you want to analyze.

For more information about SMF record formats, see the *MVS System Management Facilities (SMF)* manual.

### The SMF header

Table 16 shows the format of SMF record header (SM116).

*Table 16. SMF record header description*

| Offsets | | Type | Len | Name | Description | Example |
|---|---|---|---|---|---|---|
| Dec | Hex | | | | | |
| 0 | 0 | Structure | 28 | SM116 | SMF record header. | |
| 0 | 0 | Unsigned | 2 | SM116LEN | SMF record length. | 01A4 |
| 2 | 2 | | 2 | | Reserved. | 0000 |
| 4 | 4 | Unsigned | 1 | SM116FLG | System indicator. | 5E |
| 5 | 5 | Unsigned | 1 | SM116RTY | Record type. The SMF record type, for MQSeries accounting records this is always 116 (X'74'). | 74 |
| 6 | 6 | Unsigned | 4 | SM116TME | Time when SMF moved record. | 00356124 |
| 10 | A | Unsigned | 4 | SM116DTE | Date when SMF moved record. | 0100223F |

*Table 16. SMF record header description (continued)*

| Offsets | | Type | Len | Name | Description | Example |
|---|---|---|---|---|---|---|
| Dec | Hex | | | | | |
| 14 | E | Character | 4 | SM116SID | OS/390 subsystem ID. Defines the OS/390 subsystem on which the records were collected. | D4E5F4F1 (MV41) |
| 18 | 12 | Character | 4 | SM116SSI | MQSeries subsystem ID. | D4D8F0F7 (MQ07) |
| 22 | 16 | Unsigned | 2 | SM116STF | Record subtype. | 0000 |
| 24 | 18 | | 4 | | Reserved. | 00000000 |
| 28 | 1C | Character | 0 | SM116END | End of SMF header and start of self-defining section. | |
| **Note:** The (hexadecimal) values in the right-hand column relate to Figure 21. | | | | | | |

## Self-defining sections

A self-defining section of an SMF record tells you where to find an accounting record, how long it is, and how many times that type of record is repeated (with different values). The self-defining sections follow the header, at a fixed offset from the start of the SMF record.

Each self-defining section points to accounting related data. Table 17 summarizes the offsets from the start of the SMF record header.

*Table 17. Offsets to self-defining sections.* Offsets are from the start of the SMF record and are fixed for each type of accounting source.

| Record subtype (SMF116STF) | Source of accounting data | Offset | See... |
|---|---|---|---|
| 0 | Message manager | X'24' | “Message manager data records” on page 130 |
| | Message manager | X'2C' | |
| 1 | Thread identification record | X'24' | “Thread-level and queue-level data records” on page 132 |
| | Thread-level accounting | X'2C' | |
| | Queue-level accounting | X'34' | |
| 2 | Thread identification record | X'24' | |
| | Queue-level accounting | X'2C' | |
| **Note:** Other self-defining sections refer to data for IBM use only. | | | |

Each self-defining record is two fullwords long and has this format:

```
    ssssssssllllnnnn
```

where:

| | |
|---|---|
| **ssssssss** | Fullword containing the offset from start of the SMF record. |
| **llll** | Halfword giving the length of this data record. |
| **nnnn** | Halfword giving the number of data records in this SMF record. |

For example, in Figure 21, the self-defining section for the type of message manager accounting data is shown in bold. It is located at offset X'2C' from the start of the SMF record and contains this information:

- The offset of the message manager accounting data is located X'00000104' bytes from the start of the SMF record.
- This message manager record is X'0030' bytes long.
- There is one record (X'0001').

**Note:** Always use offsets in the self-defining sections to locate the accounting records.

## Example SMF header and self-defining section

Figure 21 shows an example of part of an SMF type 116 record. The numbers in the left-hand column represent the offset, in hexadecimal, from the start of the record. Each line corresponds to sixteen bytes of data, where each byte is two hexadecimal characters, for example 0C. The characters in the right-hand column represent the printable characters for each byte. Non-printable characters are shown by a period (.) character.

In this example, alternate fields in the SMF header are underlined to help you to see them; refer to Table 16 to identify them. The self defining section for one of the message manager accounting data records (at the offset given in Table 17 on page 128) is shown in bold.

```
00000000. 01A40000 5E740035 61240100 223FD4E5 |....;.../.....MV|
00000010. F4F1D4D8 F0F70000 00000000 00000134 |41MQ07..........|
00000020. 00700001 00000054 00B00001 00000104 |................|
00000030. 00300001 00000000 00000000 00000000 |................|
00000040. 00000000 00000000 00000000 00000000 |................|
```

*Figure 21. Part of an SMF record showing the header and self-defining sections*

## Processing type 116 SMF records

Any accounting data you collect from SMF must be processed in order to extract useful information. When you process the data, verify that the records are from MQSeries and that they are the records you are expecting.

Validate the value of the following fields:
- SM116RTY, the SMF record number = X'74' (116)
- SM116STF, the record subtype, must be 0000, 0001, or 0002

A program that you can use to format and print MQSeries accounting records is supplied in an MQSeries SupportPac.

## Message manager data records

| The message manager is the component of MQSeries that processes all MQI
| requests. The format of the message manager accounting records are described in
| assembler macros thlqual.SCSQMACS(CSQDQWHC),
| thlqual.SCSQMACS(CSQDQWHS), and thlqual.SCSQMACS(CSQDQMAC).

### Interpretation

The QMAC data gives you information about the CPU time spent processing MQSeries calls, and counts of the number of **MQPUT** and **MQGET** requests for messages of different sizes.

The QWHC data gives you information about the user (for example, the user ID (QWHCAID) and the type of application (QWHCATYP)).

**Note:** A single IMS application might write two SMF records. In this case, the figures from both records should be added to provide the correct totals for the IMS application.

#### Records containing zero CPU time

Records are sometimes produced that contain zero CPU time in the QMACCPUT field. These records occur when long running TCBs identified to MQSeries either terminate or are prompted to output accounting records by accounting trace being stopped. Such TCBs exist in the CICS adapter and in the channel initiator (for distributed queuing without CICS). The number of these TCBs with zero CPU time depends upon how much activity there has been in the system:

- For the CICS adapter, this can result in up to nine records with zero CPU time.
- For the channel initiator, the number of records with zero CPU time can be up to the sum of `Adapters + Dispatchers + 6`, as defined in the channel initiator parameters.

These records reflect the amount of work done under the TCB, and can be ignored.

#### Thread cross reference data

The interpretation of the data in the thread cross reference (QWHCCV) field varies. This depends on what the data relates to:

- CICS (QWHCATYP=1) – see Table 18
- IMS (QWHCATYP=3 or 4) – see Table 19 on page 131
- Batch, TSO, or RRS Batch (QWHCATYP=2 or 7) – this field consists of binary zeros
- Others – no meaningful data

*Table 18. Structure of the thread cross reference record for a CICS system*

| Offsets | | Type | Len | Name | Description |
|---|---|---|---|---|---|
| Dec | Hex | | | | |
| 48 | (30) | Character | 4 | QWHCTNO | CICS thread number. |
| 52 | (34) | Character | 4 | QWHCTRN | CICS transaction name. |
| 56 | (38) | Signed | 4 | QWHCTASK | CICS task number. |

Some entries contain blank characters. These apply to the TCB, rather than to a specific transaction.

*Table 19. Structure of the thread cross reference record for an IMS system*

| **Offsets** | | **Type** | **Len** | **Name** | **Description** |
|---|---|---|---|---|---|
| **Dec** | **Hex** | | | | |
| 48 | (30) | Character | 4 | QWHCPST | IMS partition specification table (PST) region identifier. |
| 52 | (34) | Character | 8 | QWHCPSB | IMS program specification block (PSB) name. |

## Sample subtype 0 accounting record

Figure 22 shows a type 116, subtype 0 SMF record. In this figure, the SMF record header, the self-defining section, and the QMAC accounting data record are underlined.

```
000000  01A40000 5E740035 61240100 223FD4E5   *....;.../.....MV*
000010  F4F1D4D8 F0F70000 00000000 00000134   *41MQ07..........*
000020  00700001 00000054 00B00001 00000104   *................*
000030  00300001 00000000 00000000 00000000   *................*
000040  00000000 00000000 00000000 00000000   *................*
000050  00000000 B478AB43 9C6C2280 B478AB47   *.........%......*
000060  9DB47E02 00000000 04C0F631 00000001   *..=......{6.....*
000070  9880E72D 00000000 014D9540 00000000   *..X......(. ....*
000080  08480C80 00000010 40404040 40404040   *........        *
000090  00000000 00000000 00000051 00000000   *................*
0000A0  00000000 00000000 00000000 00000000   *................*
0000B0  00000000 00000000 00000000 00000000   *................*
0000C0  00000000 00000000 00000000 00000000   *................*
0000D0  00000000 00000000 00000000 00000000   *................*
0000E0  00000000 00000000 00000000 00000000   *................*
0000F0  00000000 00000000 00000000 00000000   *................*
000100  00000000 D4140030 D8D4C1C3 00000000   *....M...QMAC....*
000110  689C738D 00000050 00000000 00000050   *.......&.......&*
000120  0000000A 00000000 00000000 00000000   *................*
000130  00000000 0024011A 00030710 02DAACF0   *...............0*
```

*Figure 22. Example SMF type 116, subtype 0 record*

## Thread-level and queue-level data records

Thread level accounting records are collected for each task using MQSeries. For each task, a thread-level accounting data record is written to the SMF when the task finishes. For a long running task, data is also written at the statistics interval set by the STATIME parameter of the CSQ6SYSP system parameter macro (or by the system SMF statistics broadcast), provided that the task was running the previous time statistics were gathered. In addition, accounting information is gathered about each queue that the task opens. A queue-level accounting record is written for each queue that the task has used since the thread-level accounting record was last written.

Thread-level and queue-level accounting records are produced if you specify class 3 when you start the accounting trace (using the START TRACE command described in the *MQSeries MQSC Command Reference*).

The thread level accounting information is written to an SMF type 116, subtype 1 record, and is followed by up to 48 queue-level records. If the task opened more than 48 queues, further queue information is written to one or more SMF type 116 subtype 2 records. A thread identification control block is included in each subtype 1 and 2 record to enable you to relate each record to the correct task.

### Interpretation

The format of the thread-level accounting record is described in assembler macro thlqual.SCSQMACS(CSQDWTAS). The format of the queue-level accounting record is described in assembler macro thlqual.SCSQMACS(CSQDWQ). The format of the thread identification record is described in assembler macro thlqual.SCSQMACS(CSQDWTID).

MQSeries SupportPac MP1B (MQSeries for OS/390 V5.2 - Interpreting accounting and statistics data) describes how to interpret this data.

### Sample subtype 1 and subtype 2 records

Figure 23 and Figure 24 on page 133 show examples of SMF type 116, subtype 1 and subtype 2 records. These two accounting records were created for a batch job that opened 80 queues. Because more than 48 queues were opened, a subtype 2 record was required to contain all the information produced.

```
000000  703C0000 5E74002D 983B0100 229FD4E5  *....;.........MV*
000010  F4F1D4D8 F0F70001 00000000 00006FCC  *41MQ07........?.*
000020  00700001 0000003C 00D00001 0000010C  *.........}......*
000030  02C00001 000003CC 02400030 F70000D0  *.{....... ..7..}*
000040  E6E3C9C4 00000000 00000000 00000040  *WTID........... *
.
.
.
000100  00000000 00000000 7F4A4BB8 F70102C0  *........"...7..{*
000110  E6E3C1E2 B4802373 0BF07885 7F4AE718  *WTAS.....0..".X.*
```

*Figure 23. Example SMF type 116, subtype 1 record.* This record contains a CSQDWTID control block, the CSQDWTAS control block, and 48 CSQDWQST control blocks.

The first self-defining block starts at X'24' and is highlighted in the example; X'0000003C' is the offset to the WTID data record, X'00D0' is the length of the WTID record, and X'0001' is the number of WTID records.

The second self-defining block starts at X'2C'; X'0000010C' is the offset to the WTAS data record, X'02C0' is the length of the WTAS record, and X'0001' is the number of WTAS records.

The third self-defining block starts at X'34' and is highlighted in the example; X'000003CC' is the offset to the first WQST data record, X'0240' is the length of the WQST record, and X'0030' is the number of WQST records.

Figure 24 shows an example of an SMF type 116, subtype 2 record.

```
000000  49740000 5E74002D 983B0100 229FD4E5  *....;.........MV*
000010  F4F1D4D8 F0F70002 00000000 00004904  *41MQ07..........*
000020  00700001 00000034 00D00001 00000104  *.........}......*
000030  02400020 F70000D0 E6E3C9C4 00000002  *. ..7..}WTID....*
.
.
.
000100  7F4A4BB8 F7020240 E6D8E2E3 00000001  *"...7.. WQST....*
```

*Figure 24. Example SMF type 116, subtype 2 record.* This record contains a CSQDWTID control block and 32 CSQDWQST control blocks.

The first self-defining block starts at X'24' and is highlighted in the example; X'00000034' is the offset to WTID data record, X'00D0' is the length of the WTID record, and X'0001' is the number of WTID records.

The second self-defining block starts at X'2C'; X'00000104' is the offset to first WQST data record, X'0240' is the length of the WQST record, and X'0020' is the number of WQST records.

# Part 5. Setting up security

# Chapter 12. Using RACF classes and profiles

This chapter discusses the following subjects:
- "Using RACF security classes"
- "RACF profiles" on page 138
- "Switch profiles" on page 139

## Using RACF security classes

RACF classes are used to hold the profiles required for MQSeries security checking. Each RACF class holds one or more profiles used at some point in the checking sequence, as shown in Table 20.

*Table 20. RACF classes used by MQSeries*

| **Member class** | **Group class** | **Contents** |
|---|---|---|
| MQADMIN | GMQADMIN | Profiles:<br><br>Used mainly for holding profiles for administration-type functions. For example:<br>• Profiles for MQSeries security switches<br>• The RESLEVEL security profile<br>• Profiles for alternate user security<br>• The context security profile<br>• Profiles for command resource security |
| MQCONN | | Profiles used for connection security |
| MQCMDS | | Profiles used for command security |
| MQQUEUE | GMQQUEUE | Profiles used in queue resource security |
| MQPROC | GMQPROC | Profiles used in process resource security |
| MQNLIST | GMQNLIST | Profiles used in namelist resource security |

Some classes have a related *group class* that enables you to put together groups of resources that have similar access requirements. For details about the difference between the member and group classes and when to use a member or group class, see the *Security Server (RACF) Security Administrator's Guide*.

The classes must be activated before security checks can be made. To activate all of the MQSeries classes, you use can use this RACF command:

```
SETROPTS CLASSACT(MQADMIN,MQQUEUE,MQPROC,MQNLIST,MQCONN,MQCMDS)
```

You should also ensure that you set up the classes so that they can accept generic profiles. You also do this with the RACF command SETROPTS, for example:

```
SETROPTS GENERIC(MQADMIN,MQQUEUE,MQPROC,MQNLIST,MQCONN,MQCMDS)
```

## RACF profiles

All RACF profiles used by MQSeries contain a prefix. For queue-sharing group level security, this is the queue-sharing group name. For queue manager level security, the prefix is the queue manager name. If you are using a mixture of queue manager and queue-sharing group level security, you will use profiles with both types of prefix. (Queue-sharing group and queue manager level security are described in the *MQSeries for OS/390 Concepts and Planning Guide*.)

For example, if you want to protect a queue called QUEUE_FOR_SUBSCRIBER_LIST in queue-sharing group QSG1 at queue-sharing group level, the appropriate profile would be defined to RACF as:

```
RDEFINE MQQUEUE QSG1.QUEUE_FOR_SUBSCRIBER_LIST
```

If you want to protect a queue called QUEUE_FOR_LOST_CARD_LIST, that belongs to queue manager STCD at queue manager level, the appropriate profile would be defined to RACF as:

```
RDEFINE MQQUEUE STCD.QUEUE_FOR_LOST_CARD_LIST
```

This means that different queue managers and queue-sharing groups can share the same RACF database and yet have different security options.

Do not use generic queue manager names in profiles in order to avoid unanticipated user access.

MQSeries allows the use of the percent character (%) in object names. However, RACF uses the % character as a single-character wild card. This means that when you define an object name with a % character in its name, you must consider this when you define the corresponding profile.

For example, for the queue CREDIT_CARD_%_RATE_INQUIRY, on queue manager CRDP, the profile would be defined to RACF as follows:

```
RDEFINE MQQUEUE CRDP.CREDIT_CARD_%_RATE_INQUIRY
```

This queue cannot be protected by a generic profile, such as, CRDP.**.

## Switch profiles

To control the security checking performed by MQSeries, you must define *switch profiles*. A switch profile is a normal RACF profile that has a special meaning to MQSeries. The access list in switch profiles is not used by MQSeries.

MQSeries maintains an internal switch for each of the switch types shown in tables 21 through 24. Switch profiles can be maintained at queue-sharing group level or at queue manager level or at a combination of both. Using a single set of queue-sharing group security switch profiles, you can control security on all the queue managers within a queue-sharing group.

When a security switch is set on, the security checks associated with the switch are performed. When a security switch is set off, the security checks associated with the switch are bypassed. The default is that all security switches are set on.

### Switches and classes

When a queue manager is started (or when the MQADMIN class is refreshed by the MQSeries REFRESH SECURITY command), MQSeries first checks the status of RACF and the MQADMIN class. It sets the subsystem security switch off if it discovers one of these conditions:

- RACF is inactive or not installed.
- The MQADMIN class is not defined (this class is always defined for RACF because it is included in the class descriptor table (CDT)).
- The MQADMIN class has not been activated.

If both RACF and the MQADMIN class are active, MQSeries checks the MQADMIN class to see whether any of the switch profiles have been defined. It first checks the profiles described in “Profiles to control subsystem security” on page 140. If subsystem security is not required, MQSeries sets the internal subsystem security switch off, and performs no further checks.

The profiles determine whether the corresponding MQSeries switch is set off and that type of security is deactivated. If any MQSeries switch is set on, MQSeries checks the status of the RACF class associated with the type of security corresponding to the MQSeries switch. If the class is not installed or not active, the MQSeries switch is set off. For example, process security checks are not carried out if the MQPROC class has not been activated. The class not being active is equivalent to defining NO.PROCESS.CHECKS profile for every queue manager and queue-sharing group that uses this RACF database.

### How switches work

To set a security switch off, you need to define a NO.* switch profile for it. The existence of a NO.* profile means that security checks are **not** performed for that type of resource, unless you choose to override a queue-sharing group level setting on a particular queue manager. This is described in “Overriding queue-sharing group level settings” on page 140.

If your queue manager is not a member of a queue-sharing group, you do not need to define any queue-sharing group level profiles or any override profiles. However, you must remember to define these profiles if the queue manager joins a queue-sharing group at a later date.

### Switch profiles

Each NO.* switch profile that MQSeries detects turns off the checking for that type of resource. Switch profiles are activated during startup of the queue manager. If you change the switch profiles while any affected queue managers are running, you can get MQSeries to recognize the changes by issuing the MQSeries REFRESH SECURITY command.

The switch profiles must always be defined in the MQADMIN class. Do not define them in the GMQADMIN class. Tables 21 through 24 show the valid switch profiles and the security type they control.

### Overriding queue-sharing group level settings

You can override queue-sharing group level security settings for a particular queue manager that is a member of that group. If you want to perform queue manager checks on an individual queue manager that are not performed on other queue managers in the group, use the (qmgr-name.YES.*) switch profiles.

Conversely, if you do not want to perform a certain check on one particular queue manager within a queue-sharing group, define a (qmgr-name.NO.*) profile for that particular resource type on the queue manager, and do not define a profile for the queue-sharing group. (MQSeries only checks for a queue-sharing group level profile if it does not find a queue manager level profile.)

## Profiles to control subsystem security

The first security check made by MQSeries is used to determine whether security checks are required for the whole MQSeries subsystem. If you specify that you do not want subsystem security, no further checks are made.

The following switch profiles are checked to determine whether subsystem security is required. Figure 25 on page 141 shows the order in which they are checked.

*Table 21. Switch profiles for subsystem level security*

| **Switch profile name** | **Type of resource or checking that is controlled** |
|---|---|
| qmgr-name.NO.SUBSYS.SECURITY | Subsystem security for this queue manager |
| qsg-name.NO.SUBSYS.SECURITY | Subsystem security for this queue-sharing group |
| qmgr-name.YES.SUBSYS.SECURITY | Subsystem security override for this queue manager |

If your queue manager is not a member of a queue-sharing group, MQSeries checks for the qmgr-name.NO.SUBSYS.SECURITY switch profile only.

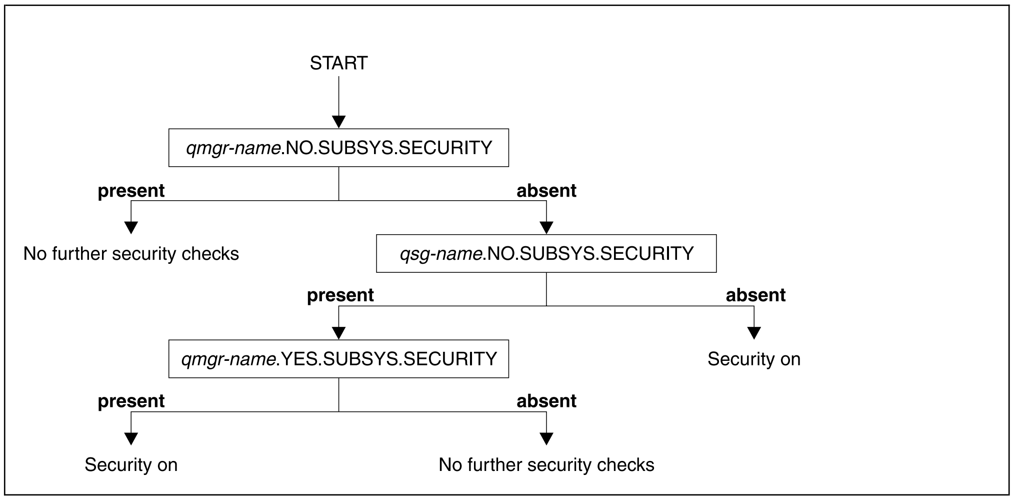
START
qmgr-name.NO.SUBSYS.SECURITY
present
absent
No further security checks
qsg-name.NO.SUBSYS.SECURITY
present
absent
qmgr-name.YES.SUBSYS.SECURITY
Security on
present
absent
Security on
No further security checks

*Figure 25. Checking for subsystem security*

## Profiles to control queue-sharing group or queue manager level security

When MQSeries has determined that security checking is required, it then determines whether checking is required at queue-sharing group or queue manager level, or both. These checks are not performed if your queue manager is not a member of a queue sharing group.

The following switch profiles are checked to determine the level required. Figure 26 on page 142 and Figure 27 on page 142 show the order in which they are checked.

*Table 22. Switch profiles for queue-sharing group or queue manager level security*

| Switch profile name | Type of resource or checking that is controlled |
|---|---|
| qmgr-name.NO.QMGR.CHECKS | No queue manager level checks for this queue manager |
| qsg-name.NO.QMGR.CHECKS | No queue manager level checks for this queue-sharing group |
| qmgr-name.YES.QMGR.CHECKS | Queue manager level checks override for this queue manager |
| qmgr-name.NO.QSG.CHECKS | No queue-sharing group level checks for this queue manager |
| qsg-name.NO.QSG.CHECKS | No queue-sharing group level checks for this queue-sharing group |
| qmgr-name.YES.QSG.CHECKS | Queue-sharing group level checks override for this queue manager |

If subsystem security is active, you cannot switch off both queue-sharing group and queue manager level security. If you try to do this, MQSeries sets security checking on at both levels.

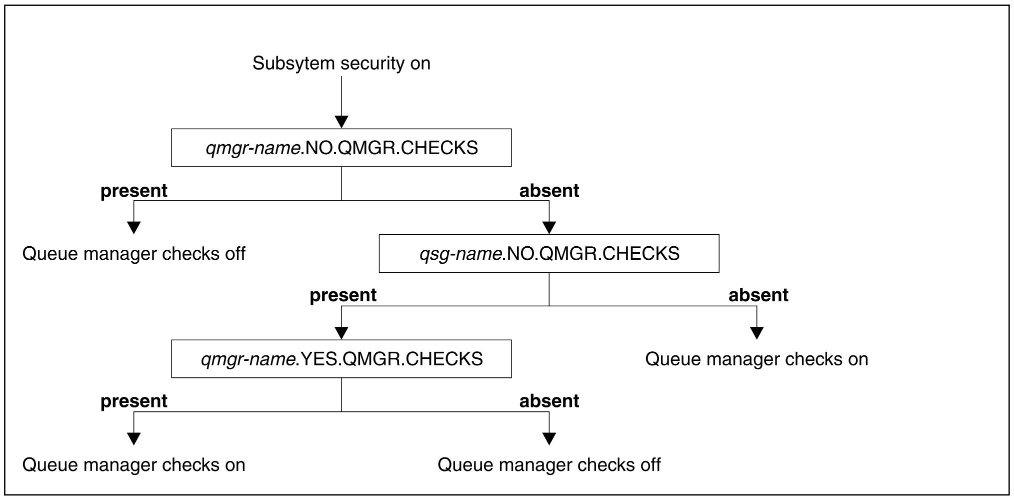

*Figure 26. Checking for queue manager level security*

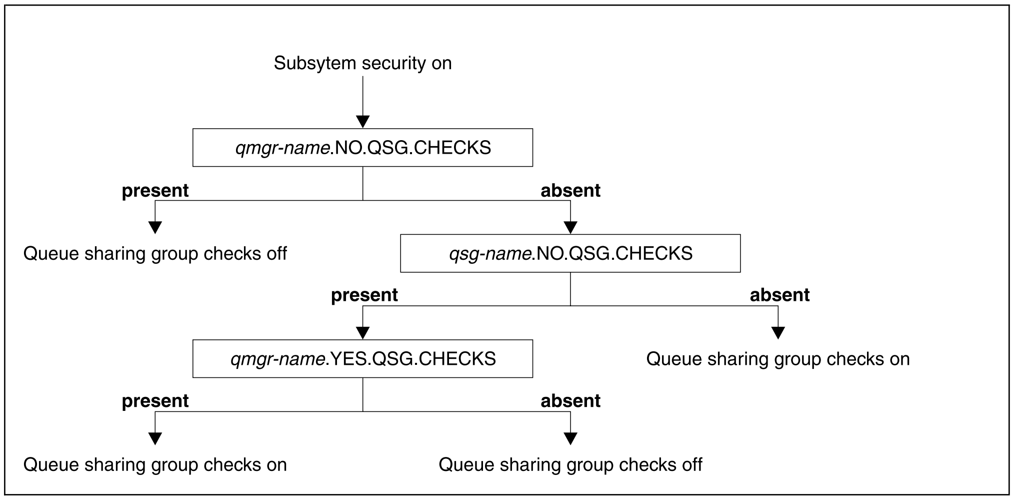

*Figure 27. Checking for queue-sharing group level security*

## Valid combinations of switches

Table 23 shows the combinations of switch settings that are valid. If you use a combination of switch settings that is not valid, message CSQH026I is issued and security checking is set on at both queue-sharing group and queue manager level.

*Table 23. Valid security switch combinations*

| Security level | Switch settings |
|---|---|
| Queue manager level security | qmgr-name.NO.QSG.CHECKS |
| | qsg-name.NO.QSG.CHECKS |
| | qmgr-name.NO.QSG.CHECKS<br>qsg-name.NO.QMGR.CHECKS<br>qmgr-name.YES.QMGR.CHECKS |
| | qsg-name.NO.QSG.CHECKS<br>qsg-name.NO.QMGR.CHECKS<br>qmgr-name.YES.QMGR.CHECKS |

*Table 23. Valid security switch combinations (continued)*

| **Security level** | **Switch settings** |
|---|---|
| Queue-sharing group level security | qmgr-name.NO.QMGR.CHECKS |
| | qsg-name.NO.QMGR.CHECKS |
| | qmgr-name.NO.QMGR.CHECKS<br>qsg-name.NO.QSG.CHECKS<br>qmgr-name.YES.QSG.CHECKS |
| | qsg-name.NO.QMGR.CHECKS<br>qsg-name.NO.QSG.CHECKS<br>qmgr-name.YES.QSG.CHECKS |
| Queue manager and queue-sharing group level security | qsg-name.NO.QMGR.CHECKS<br>qmgr-name.YES.QMGR.CHECKS<br>No QSG.* profiles defined |
| | No QMGR.* profiles defined<br>qsg-name.NO.QSG.CHECKS<br>qmgr-name.YES.QSG.CHECKS |
| | qsg-name.NO.QMGR.CHECKS<br>qmgr-name.YES.QMGR.CHECKS<br>qsg-name.NO.QSG.CHECKS<br>qmgr-name.YES.QSG.CHECKS |
| | No profiles for either switch defined |
| Other combinations that cause both levels of checking to be switched **on**. | qmgr-name.NO.QMGR.CHECKS<br>qmgr-name.NO.QSG.CHECKS |
| | qsg-name.NO.QMGR.CHECKS<br>qsg-name.NO.QSG.CHECKS |
| | qmgr-name.NO.QMGR.CHECKS<br>qsg-name.NO.QSG.CHECKS |
| | qsg-name.NO.QMGR.CHECKS<br>qmgr-name.NO.QSG.CHECKS |

## Resource level checks

Table 24 shows the switch profiles used to control access to MQSeries resources.

If your queue manager is part of a queue sharing group and you have both queue manager and queue-sharing group security active, you can use a YES.* switch profile to override queue-sharing group level profiles and specifically turn on security for a particular queue manager.

Some profiles apply to both queue managers and queue-sharing groups. These are prefixed by the string *hlq* in this book and you should substitute the name of your queue-sharing group or queue manager, as applicable. Profile names shown prefixed by *qmgr-name* are queue-manager override profiles; you should substitute the name of your queue manager.

*Table 24. Switch profiles for resource checking*

| Type of resource checking that is controlled | Switch profile name | Override profile for a particular queue manager |
|---|---|---|
| Connection security | hlq.NO.CONNECT.CHECKS | qmgr-name.YES.CONNECT.CHECKS |
| Queue security | hlq.NO.QUEUE.CHECKS | qmgr-name.YES.QUEUE.CHECKS |
| Process security | hlq.NO.PROCESS.CHECKS | qmgr-name.YES.PROCESS.CHECKS |
| Namelist security | hlq.NO.NLIST.CHECKS | qmgr-name.YES.NLIST.CHECKS |
| Context security | hlq.NO.CONTEXT.CHECKS | qmgr-name.YES.CONTEXT.CHECKS |
| Alternate user security | hlq.NO.ALTERNATE.USER.CHECKS | qmgr-name.YES.ALTERNATE.USER.CHECKS |
| Command security | hlq.NO.CMD.CHECKS | qmgr-name.YES.CMD.CHECKS |
| Command resource security | hlq.NO.CMD.RESC.CHECKS | qmgr-name.YES.CMD.RESC.CHECKS |
| **Note:** Generic switch profiles such as hlq.NO.** are ignored by MQSeries | | |

For example, say you want to perform process security checks on queue manager QM01, which is a member of queue-sharing group QSG3 but you do not want to perform process security checks on any of the other queue managers in the group. Define the following switch profiles:

```
QSG3.NO.PROCESS.CHECKS
QM01.YES.PROCESS.CHECKS
```

If you want to have queue security checks performed on all the queue managers in the queue-sharing group, except QM02, define the following switch profile:

```
QM02.NO.QUEUE.CHECKS
```

(There is no need to define a profile for the queue sharing group because the checks are automatically enabled if there is no profile defined.)

## An example of defining switches

Four MQSeries subsystems have been defined:

- MQP1 (a production system)
- MQP2 (a production system)

- MQD1 (a development system)
- MQT1 (a test system)

All four queue managers are members of queue-sharing group QS01. All MQSeries RACF classes have been defined and activated.

These subsystems have different security requirements:

- The production systems require full MQSeries security checking to be active at queue-sharing group level on both systems.

  This is done by specifying the following profile:

  ```
  RDEFINE MQADMIN QS01.NO.QMGR.CHECKS
  ```

  This sets queue-sharing group level checking for all the queue managers in the queue-sharing group. You do not need to define any other switch profiles for the production queue managers because you want to check everything for these systems.
- Test queue manager MQT1 also requires full security checking. However, because you might want to change this later, security can be defined at queue-manager level so that you can change the security settings for this queue manager without affecting the other members of the queue-sharing group.

  This is done by defining the NO.QSG.CHECKS profile for MQT1 as follows:

  ```
  RDEFINE MQADMIN MQT1.NO.QSG.CHECKS
  ```
- Development queue manager MQD1 has different security requirements from the rest of the queue-sharing group. It requires only connection and queue security to be active.

  This is done by defining a `MQD1.YES.QMGR.CHECKS` profile for this queue manager, and then defining the following profiles to switch off security checking for the resources that do not need to be checked:

  ```
  RDEFINE MQADMIN MQD1.NO.CMD.CHECKS
  RDEFINE MQADMIN MQD1.NO.CMD.RESC.CHECKS
  RDEFINE MQADMIN MQD1.NO.PROCESS.CHECKS
  RDEFINE MQADMIN MQD1.NO.NLIST.CHECKS
  RDEFINE MQADMIN MQD1.NO.CONTEXT.CHECKS
  RDEFINE MQADMIN MQD1.NO.ALTERNATE.USER.CHECKS
  ```

Once MQSeries is active, you can display the current security settings by issuing the MQSeries command DISPLAY SECURITY.

You can also change the switch settings when MQSeries is running by defining or deleting the appropriate switch profile in the MQADMIN class. To make the changes to the switch settings active, you must issue the MQSeries command REFRESH SECURITY for the MQADMIN class.

See “Security refreshes” on page 192 for more details about using the MQSeries commands DISPLAY SECURITY and REFRESH SECURITY.

# Chapter 13. Profiles used to control access to MQSeries resources

You must define RACF profiles to control access to MQSeries resources, in addition to the switch profiles that might have been defined. If you do not have a resource profile defined for a particular security check, and a user issues a request that would involve making that check, MQSeries denies access. You do not need to define profiles for security types relating to any security switches that you have deactivated.

This chapter discusses the following types of RACF profile:

- “Profiles for connection security”
- “Profiles for queue security” on page 150
- “Profiles for processes” on page 159
- “Profiles for namelists” on page 160
- “Profiles for alternate user security” on page 161
- “Profiles for context security” on page 163
- “Profiles for command security” on page 165
- “Profiles for command resource security” on page 168

## Profiles for connection security

If connection security is active, you must define profiles in the MQCONN class and permit the necessary groups or user IDs access to those profiles, so that they can connect to MQSeries subsystems.

To enable a connection to be made, you must grant users RACF READ access to the appropriate profile.

If no queue manager level profile exists, and your queue manager is a member of a queue-sharing group, checks are made against queue-sharing group level profiles.

A connection profile qualified with a queue manager name controls access to a specific queue manager and users given access to this profile can connect to that queue manager. A connection profile qualified with queue-sharing group name controls access to all queue managers within the queue-sharing group for that connection type. For example, a user with access to `QS01.BATCH` can Batch connect to any queue manager in queue-sharing group QS01 that has not got a queue manager level profile defined.

**Notes:**

1. For information about the user IDs checked for different security requests, see “Chapter 15. User IDs for security checking” on page 181.
2. Resource level security (RESLEVEL) checks are also made at connection time. For details, see “Chapter 14. Using the RESLEVEL security profile” on page 171.

## Connection security profiles for the Batch/TSO adapter

Profiles for checking connections from Batch or TSO take the form:

```
hlq.BATCH
```

where hlq can be either the qmgr-name (queue manager name) or qsg-name (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails.

RRS uses the Batch/TSO adapter for connection security.

For connection requests through Batch or TSO, you must permit the job or TSO user ID to access the connection profile. For example, the following RACF command allows Batch and TSO users in the CONNTQM1 group to connect to the queue manager TQM1:

```
RDEFINE MQCONN TQM1.BATCH UACC(NONE)
PERMIT TQM1.BATCH CLASS(MQCONN) ID(CONNTQM1) ACCESS(READ)
```

## Connection security profiles for the CICS adapter

Profiles for checking connections from CICS take the form:

```
hlq.CICS
```

where hlq can be either qmgr-name (queue manager name) or qsg-name (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails

For connection requests by CICS, you need only permit the CICS address space user ID access to the connection profile.

For example, the following RACF commands allow the CICS address space user ID KCBCICS to connect to the queue manager TQM1:

```
RDEFINE MQCONN TQM1.CICS UACC(NONE)
PERMIT TQM1.CICS CLASS(MQCONN) ID(KCBCICS) ACCESS(READ)
```

## Connection security profiles for the IMS adapter

Profiles for checking connections from IMS take the form:

```
hlq.IMS
```

where hlq can be either qmgr-name (queue manager name) or qsg-name (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails

For connection requests by IMS, permit access to the connection profile for the IMS control and dependent region user IDs.

For example, the following RACF commands allow:
- The IMS region user ID, IMSREG, to connect to the queue manager TQM1.
- Users in group BMPGRP to submit BMP jobs.

```
RDEFINE MQCONN TQM1.IMS UACC(NONE)
PERMIT TQM1.IMS CLASS(MQCONN) ID(IMSREG,BMPGRP) ACCESS(READ)
```

## Connection security profiles for distributed queuing

Profiles for checking connections from distributed queuing (without CICS ISC) take the form:

```
hlq.CHIN
```

where hlq can be either qmgr-name (queue manager name) or qsg-name (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails

For connection requests by the channel initiator, define access to the connection profile for the user ID used by the channel initiator started task address space.

For example, the following RACF commands allow the channel initiator address space running with user ID DQCTRL to connect to the queue manager TQM1:

```
RDEFINE MQCONN TQM1.CHIN UACC(NONE)
PERMIT TQM1.CHIN CLASS(MQCONN) ID(DQCTRL) ACCESS(READ)
```

## Profiles for queue security

If queue security is active, you must define profiles in the MQQUEUE or GMQQUEUE classes and permit the necessary groups or user IDs access to these profiles, so they can issue MQSeries API requests that use queues.

Profiles for queue security take the form:

```
hlq.queuename
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `queuename` is the name of the queue being opened, as specified in the object descriptor on the **MQOPEN** or **MQPUT1** call.

A profile prefixed by the queue manager name controls access to a single queue on that queue manager. A profile prefixed by the queue-sharing group name controls access to access to one or more queues with that queue name on all queue managers within the queue-sharing group, or access to a shared queue by any queue manager within the group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that queue on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

If you are using shared queues, you are recommended to use queue-sharing group level security.

For details of how queue security operates when the queue name is that of an alias or a model queue, see “Considerations for alias queues” on page 152 and “Considerations for model queues” on page 153.

The RACF access required to open a queue depends on the **MQOPEN** or **MQPUT1** options specified. If more than one of the MQOO_* and MQPMO_* options is coded, the queue security check is performed for the highest RACF authority required.

*Table 25. Access levels for queue security*

| MQOPEN or MQPUT1 option | RACF access level required to access hlq.queuename |
|---|---|
| MQOO_BROWSE | READ |
| MQOO_INQUIRE | READ |
| MQOO_BIND_* | UPDATE |
| MQOO_INPUT_* | UPDATE |
| MQOO_OUTPUT or MQPUT1 | UPDATE |
| MQOO_PASS_ALL_CONTEXT<br>MQPMO_PASS_ALL_CONTEXT | UPDATE |
| MQOO_PASS_IDENTITY_CONTEXT<br>MQPMO_PASS_IDENTITY_CONTEXT | UPDATE |
| MQOO_SAVE_ALL_CONTEXT | UPDATE |

*Table 25. Access levels for queue security (continued)*

| MQOPEN or MQPUT1 option | RACF access level required to access hlq.queuename |
|---|---|
| MQOO_SET_IDENTITY_CONTEXT<br>MQPMO_SET_IDENTITY_CONTEXT | UPDATE |
| MQOO_SET_ALL_CONTEXT<br>MQPMO_SET_ALL_CONTEXT | UPDATE |
| MQOO_SET | ALTER |

For example, on MQSeries subsystem QM77, all user IDs in the RACF group PAYGRP are to be given access to get messages from or put messages to all queues with names beginning with 'PAY.'. You can do this using these RACF commands:

```
RDEFINE MQQUEUE QM77.PAY.** UACC(NONE)
PERMIT QM77.PAY.** CLASS(MQQUEUE) ID(PAYGRP) ACCESS(UPDATE)
```

Also, all user IDs in the PAYGRP group must have access to put messages on queues that do not follow the PAY naming convention. For example:

```
REQUEST_QUEUE_FOR_PAYROLL
SALARY.INCREASE.SERVER
REPLIES.FROM.SALARY.MODEL
```

You can do this by defining profiles for these queues in the GMQQUEUE class and giving access to that class as follows:

```
RDEFINE GMQQUEUE PAYROLL.EXTRAS UACC(NONE)
        ADDMEM(QM77.REQUEST_QUEUE_FOR_PAYROLL,
               QM77.SALARY.INCREASE.SERVER,
               QM77.REPLIES.FROM.SALARY.MODEL)
PERMIT PAYROLL.EXTRAS CLASS(GMQQUEUE) ID(PAYGRP) ACCESS(UPDATE)
```

**Notes:**

1. If the RACF access level that an application has to a queue security profile is changed, the changes will only take effect for any new object handles obtained (that is, new **MQOPEN**s) for that queue. Those handles already in existence at the time of the change retain their existing access to the queue. If an application is required to use its changed access level to the queue rather than its existing access level, it must close and re-open the queue for each object handle that requires the change.
2. In the example, the subsystem name QM77 could also be the name of a queue-sharing group.

Other types of security checks might also occur at the time the queue is opened depending on the open options specified and the types of security that are active. See also “Profiles for context security” on page 163 and “Profiles for alternate user security” on page 161. For a summary table showing the open options and the security authorization needed when queue, context, and alternate user security are all active, see Table 29 on page 157.

## Considerations for alias queues

When you issue an **MQOPEN** or **MQPUT1** call for an alias queue, MQSeries makes a resource check against the queue name specified in the object descriptor (MQOD) on the call. It does not check if the user is allowed access to the target queue name.

For example, an alias queue called PAYROLL.REQUEST resolves to a target queue of PAY.REQUEST. If queue security is active, you need only be authorized to access the queue PAYROLL.REQUEST. No check is made to see if you are authorized to access the queue PAY.REQUEST.

## Using alias queues to distinguish between MQGET and MQPUT requests

The range of MQI calls available in one access level can cause a problem if you want to restrict access to a queue to allow only the **MQPUT** call or only the **MQGET** call. A queue can be protected by defining two aliases that resolve to that queue: one that enables applications to get messages from the queue, and one that enable applications to put messages on the queue.

The following text gives you an example of how you can define your queues to MQSeries:

```
DEFINE QLOCAL(MUST_USE_ALIAS_TO_ACCESS) GET(ENABLED)
       PUT(ENABLED)

DEFINE QALIAS(USE_THIS_ONE_FOR_GETS) GET(ENABLED)
       PUT(DISABLED) TARGQ(MUST_USE_ALIAS_TO_ACCESS)

DEFINE QALIAS(USE_THIS_ONE_FOR_PUTS) GET(DISABLED)
       PUT(ENABLED) TARGQ(MUST_USE_ALIAS_TO_ACCESS)
```

You must also make the following RACF definitions:

```
RDEFINE MQQUEUE hlq.MUST_USE_ALIAS_TO_ACCESS UACC(NONE)
RDEFINE MQQUEUE hlq.USE_THIS_ONE_FOR_GETS UACC(NONE)
RDEFINE MQQUEUE hlq.USE_THIS_ONE_FOR_PUTS UACC(NONE)
```

Then you ensure that no users have access to the queue hlq.MUST_USE_ALIAS_TO_ACCESS, and give the appropriate users or groups access to the alias. You can do this using the following RACF commands:

```
PERMIT hlq.USE_THIS_ONE_FOR_GETS CLASS(MQQUEUE)
       ID(GETUSER,GETGRP) ACCESS(UPDATE)
PERMIT hlq.USE_THIS_ONE_FOR_PUTS CLASS(MQQUEUE)
       ID(PUTUSER,PUTGRP) ACCESS(UPDATE)
```

This means user ID GETUSER and user IDs in the group GETGRP are only allowed to get messages on MUST_USE_ALIAS_TO_ACCESS through the alias queue USE_THIS_ONE_FOR_GETS; and user ID PUTUSER and user IDs in the group PUTGRP are only allowed to put messages through the alias queue USE_THIS_ONE_FOR_PUTS.

If you want to use a technique like this, you must inform your application developers, so they can design their programs appropriately.

## Considerations for model queues

When you open a model queue, MQSeries security makes two queue security checks:

1. Are you authorized to access the model queue?
2. Are you authorized to access the dynamic queue to which the model queue resolves?

If the dynamic queue name contains a trailing * character, this * is replaced by a character string generated by MQSeries, to create a dynamic queue with a unique name. However, because the whole name, including this generated string, is used for checking authority, you should define generic profiles for these queues.

For example, an **MQOPEN** call uses a model queue name of CREDIT.CHECK.REPLY.MODEL and a dynamic queue name of CREDIT.REPLY.* on queue manager (or queue-sharing group) MQSP.

To do this, you must issue the following RACF commands to define the necessary queue profiles:

```
RDEFINE MQQUEUE MQSP.CREDIT.CHECK.REPLY.MODEL
RDEFINE MQQUEUE MQSP.CREDIT.REPLY.**
```

You must also issue the corresponding RACF PERMIT commands to allow the user access to these profiles.

A typical dynamic queue name created by an **MQOPEN** is something like CREDIT.REPLY.A346EF00367849A0. The precise value of the last qualifier is unpredictable; this is why you should use generic profiles for such queue names.

A number of MQSeries utilities put messages on dynamic queues. You should define profiles for the following dynamic queue names, and provide RACF UPDATE access to the relevant user IDs (see “Chapter 15. User IDs for security checking” on page 181 for the correct user IDs):

```
SYSTEM.CSQUTIL.*  (used by CSQUTIL)
SYSTEM.CSQOREXX.* (used by the operations and control panels)
SYSTEM.CSQXCMD.*  (used by the channel initiator when processing CSQINPX)
CSQ4SAMP.*        (used by the MQSeries supplied samples)
```

You might also consider defining a profile to control use of the dynamic queue name used by default in the application programming copy members. The MQSeries-supplied copybooks contain a default *DynamicQName*, which is CSQ.*. This enables an appropriate RACF profile to be established.

**Note:** Do not allow application programmers to specify a single * for the dynamic queue name. If you do, you must define an hlq.** profile in the MQQUEUE class, and you would have to give it wide-ranging access. This means that this profile could also be used for other non-dynamic queues that do not have a more specific RACF profile. Your users could, therefore, gain access to queues you do not want them to access.

| 
## Close options on permanent dynamic queues

If an application opens a permanent dynamic queue that was created by another application and then attempts to delete that queue with an **MQCLOSE** option, some extra security checks are applied when the attempt is made. See Table 26.

*Table 26. Access levels for close options on permanent dynamic queues*

| **MQCLOSE option** | **RACF access level required to hlq.queuename** |
|---|---|
| MQCO_DELETE | ALTER |
| MQCO_DELETE_PURGE | ALTER |

## Security and remote queues

When a message is put on a remote queue, the queue security that is performed by the local queue manager depends on how the remote queue is specified when it is opened. For example:

1. If the remote queue has been defined on the local queue manager through the MQSeries command, DEFINE QREMOTE, the queue that is checked is the name of the remote queue. For example, if a remote queue is defined on queue manager MQS1 as follows:

```
DEFINE QREMOTE(BANK7.CREDIT.REFERENCE)
       RNAME(CREDIT.SCORING.REQUEST)
       RQMNAME(BNK7)
       XMITQ(BANK1.TO.BANK7)
```

   In this case, a profile for BANK7.CREDIT.REFERENCE, must be defined in the MQQUEUE class.
2. If the *ObjectQMgrName* for the request does not resolve to the local queue manager, the queue used for queue security is the name of the transmission queue used to send messages to the remote queue manager specified by the *MQOD_ObjectQMgrName*.

   For example, the transmission queue BANK1.TO.BANK7 is defined on queue manager MQS1. An **MQPUT1** request is then issued on MQS1 specifying *ObjectName* as BANK1.INTERBANK.TRANSFERS and an *ObjectQMgrName* of BANK1.TO.BANK7. In this case, the user performing the request must have access to MQS1.BANK1.TO.BANK7.
3. If you make an **MQPUT** request to a queue and specify *ObjectQMgrName* as the name of an alias of the local queue manager, only the queue name is checked for security, not that of the queue manager.

When the message gets to the remote queue manager it might be subject to additional security processing. For more information, see the *MQSeries Intercommunication* manual.

## Dead-letter queue security

Undelivered messages can be put on a special queue called the dead-letter queue. If you have sensitive data that could possibly end up on this queue, you must consider the security implications of this because you do not want unauthorized users to be able to retrieve this data.

Each of the following must be able to put messages onto the dead-letter queue:

- Application programs.
- The channel initiator address space and any MCA user IDs. (If the RESLEVEL profile is not present, or is defined so that network-received user IDs are checked, the network-received user ID also needs authority to put messages on the dead-letter queue.)
- For distributed queuing using CICS, the various MCA transactions.
- CKTI, the MQSeries-supplied CICS task initiator.
- CSQQTRMN, the MQSeries-supplied IMS trigger monitor.

The only application able to retrieve messages from the dead-letter queue should be a ‘special’ application that processes these messages. However, a problem arises if you give applications RACF UPDATE authority to the dead-letter queue for **MQPUT**s because they can then automatically retrieve messages from the queue using **MQGET** calls. You cannot disable the dead-letter queue for get operations because, if you do, not even the ‘special’ applications could retrieve the messages.

One solution to this problem is set up a two-level access to the dead-letter queue. CKTI, message channel agent transactions or the channel initiator address space, and ‘special’ applications have direct access; other applications can only access the dead-letter queue through an alias queue. This alias is defined to allow applications to put messages on the dead-letter queue, but not to get messages from it.

This is how it might work:

1. Define the real dead-letter queue with attributes PUT(ENABLED) and GET(ENABLED), as shown in the sample thlqual.SCSQPROC(CSQ4INYG).
2. Give RACF UPDATE authority for the dead-letter queue to the following user IDs:
   - User IDs that the CKTI and the MCAs or channel initiator address space run under.
   - The user IDs associated with the ‘special’ dead-letter queue processing application.
3. Define an alias queue that resolves to the real dead-letter queue, but give the alias queue these attributes: PUT(ENABLED) and GET(DISABLED). Give the alias queue a name with the same stem as the dead-letter queue name but append the characters “.PUT” to this stem. For example, if the dead-letter queue name is hlq.DEAD.QUEUE, the alias queue name would be hlq.DEAD.QUEUE.PUT.
4. To put a message on the dead-letter queue, an application uses the alias queue. This is what your application must do:
   - Retrieve the name of the real dead-letter queue. To do this, it opens the queue manager object using **MQOPEN** and then issues an **MQINQ** to get the dead-letter queue name.
   - Build the name of the alias queue by appending the characters ‘.PUT’ to this name, in this case, hlq.DEAD.QUEUE.PUT.

- Open the alias queue, hlq.DEAD.QUEUE.PUT.
- Put the message on the real dead-letter queue by issuing an **MQPUT** against the alias queue.

5. Give the user ID associated with the application RACF UPDATE authority to the alias, but no access (authority NONE) to the real dead-letter queue. This means that:
   - The application can put messages onto the dead-letter queue using the alias queue.
   - The application cannot get messages from the dead-letter queue using the alias queue because the alias queue is disabled for get operations.

   The application cannot get any messages from the real dead-letter queue either because it does have the correct RACF authority.

Table 27 summarizes the RACF authority required for the various participants in this solution.

*Table 27. RACF authority to the dead-letter queue and its alias*

| **Associated user IDs** | **Real dead-letter queue (hlq.DEAD.QUEUE)** | **Alias dead-letter queue (hlq.DEAD.QUEUE.PUT)** |
|---|---|---|
| MCA or channel initiator address space and CKTI | UPDATE | NONE |
| 'Special' application (for dead-letter queue processing) | UPDATE | NONE |
| User-written application user IDs | NONE | UPDATE |

If you use this method, the application cannot determine the maximum message length (MAXMSGL) of the dead-letter queue. This is because the MAXMSGL attribute cannot be retrieved from an alias queue. Therefore, your application should assume that the maximum message length is 100 MB, the maximum size MQSeries for OS/390 supports. The real dead-letter queue should also be defined with a MAXMSGL attribute of 100 MB.

**Note:** User-written application programs should not normally use alternate user authority to put messages on the dead-letter queue. This reduces the number of user IDs that have access to the dead-letter queue.

## System queue security

Many of the system queues are accessed by the ancillary parts of the queue manager:
- The CSQUTIL utility
- The operations and control panels
- The channel initiator address space
- The CICS transactions (for distributed queuing using CICS)

The user IDs under which these run must be given RACF access to these queues, as shown in Table 28 on page 157.

*Table 28. Access required to the SYSTEM queues by the queue manager*

| | CSQUTIL | Operations and control panels | Channel initiator for distributed queuing without CICS | Transactions for distributed queuing with CICS |
|---|---|---|---|---|
| SYSTEM.ADMIN.CHANNEL.EVENT | | | UPDATE | |
| SYSTEM.CHANNEL.COMMAND | | | | UPDATE |
| SYSTEM.CHANNEL.INITQ | | | UPDATE | |
| SYSTEM.CHANNEL.SEQNO | | | | UPDATE |
| SYSTEM.CHANNEL.SYNCQ | | | UPDATE | |
| SYSTEM.CLUSTER.COMMAND.QUEUE | | | ALTER | |
| SYSTEM.CLUSTER.REPOSITORY.QUEUE | | | UPDATE | |
| SYSTEM.CLUSTER.TRANSMIT.QUEUE | | | ALTER | |
| SYSTEM.COMMAND.INPUT | UPDATE | UPDATE | UPDATE | |
| SYSTEM.COMMAND.REPLY.MODEL | UPDATE | UPDATE | UPDATE | |
| SYSTEM.CSQOREXX.* | | UPDATE | | |
| SYSTEM.CSQUTIL.* | UPDATE | | | |
| \| SYSTEM.CSQXCMD.* | | | UPDATE | |
| \| SYSTEM.QSG.CHANNEL.SYNCQ | | | UPDATE | |
| \| SYSTEM.QSG.TRANSMIT.QUEUE | | | UPDATE | |

## API-resource security access quick reference

Table 29 summarizes the **MQOPEN**, **MQPUT1**, and **MQCLOSE** options and the access required by the different resource security types.

*Table 29. MQOPEN, MQPUT1, and MQCLOSE options and the security authorization required*. Callouts shown like this **(1)** refer to the notes following this table.

| | Minimum RACF access level required | | |
|---|---|---|---|
| RACF class: | MQQUEUE **(1)** | MQADMIN | MQADMIN |
| RACF profile: | **(2)** | **(3)** | **(4)** |
| **MQOPEN** option | | | |
| MQOO_INQUIRE **(1)** | READ **(5)** | No check | No check |
| MQOO_BROWSE | READ | No check | No check |
| MQOO_INPUT_★ | UPDATE | No check | No check |
| MQOO_SAVE_ALL_CONTEXT **(6)** | UPDATE | No check | No check |
| MQOO_OUTPUT (USAGE=NORMAL) **(7)** | UPDATE | No check | No check |
| MQOO_PASS_IDENTITY_CONTEXT **(8)** | UPDATE | READ | No check |
| MQOO_PASS_ALL_CONTEXT **(8) (9)** | UPDATE | READ | No check |
| MQOO_SET_IDENTITY_CONTEXT **(8) (9)** | UPDATE | UPDATE | No check |
| MQOO_SET_ALL_CONTEXT **(8) (10)** | UPDATE | CONTROL | No check |
| MQOO_OUTPUT (USAGE (XMITQ)) **(11)** | UPDATE | CONTROL | No check |
| MQOO_SET | ALTER | No check | No check |
| MQOO_ALTERNATE_USER_AUTHORITY **(1)** | **(12)** | **(12)** | UPDATE |
| **MQPUT1** option | | | |

*Table 29. MQOPEN, MQPUT1, and MQCLOSE options and the security authorization required (continued)*. Callouts shown like this **(1)** refer to the notes following this table.

| | | | |
|---|---|---|---|
| Put on a normal queue **(7)** | UPDATE | No check | No check |
| MQPMO_PASS_IDENTITY_CONTEXT | UPDATE | READ | No check |
| MQPMO_PASS_ALL_CONTEXT | UPDATE | READ | No check |
| MQPMO_SET_IDENTITY_CONTEXT | UPDATE | UPDATE | No check |
| MQPMO_SET_ALL_CONTEXT | UPDATE | CONTROL | No check |
| MQOO_OUTPUT<br><br>Put on a transmission queue **(11)** | UPDATE | CONTROL | No check |
| MQPMO_ALTERNATE_USER_AUTHORITY | **(13)** | **(13)** | UPDATE |
| **MQCLOSE** option | | | |
| MQCO_DELETE **(14)** | ALTER | No check | No check |
| MQCO_DELETE_PURGE **(14)** | ALTER | No check | No check |

**Notes:**

1. This option is not restricted to queues. Use the MQNLIST class for namelists, and the MQPROC class for processes.
2. Use RACF profile: hlq.resourcename
3. Use RACF profile: hlq.CONTEXT
4. Use RACF profile: hlq.ALTERNATE.USER.alternateuserid

   | alternateuserid is the user identifier that is specified in the *AlternateUserId* field of the object descriptor. Note that up to 12 characters of the *AlternateUserId* field are used for this check, unlike other checks where only the first 8 characters of a user identifier are used.
5. No check is made when opening the queue manager for inquiries.
6. MQOO_INPUT_★ must be specified as well. This is valid for a local, model or alias queue.
7. This check is done for a local or model queue that has a *Usage* queue attribute of MQUS_NORMAL, and also for an alias or remote queue (that is defined to the connected queue manager.) If the queue is a remote queue that is opened specifying an *ObjectQMgrName* (not the name of the connected queue manager) explicitly, the check is carried out against the queue with the same name as *ObjectQMgrName* (which must be a local queue with a *Usage* queue attribute of MQUS_TRANSMISSION).
8. MQOO_OUTPUT must be specified as well.
9. MQOO_PASS_IDENTITY_CONTEXT is implied as well by this option.
10. MQOO_PASS_IDENTITY_CONTEXT, MQOO_PASS_ALL_CONTEXT and MQOO_SET_IDENTITY_CONTEXT are implied as well by this option.
11. This check is done for a local or model queue that has a *Usage* queue attribute of MQUS_TRANSMISSION, and is being opened directly for output. It does not apply if a remote queue is being opened.
12. At least one of MQOO_INQUIRE, MQOO_BROWSE, MQOO_INPUT_★, MQOO_OUTPUT or MQOO_SET must be specified as well. The check carried out is the same as that for the other options specified.
13. The check carried out is the same as that for the other options specified.
14. | This only applies for permanent dynamic queues that have been opened directly, that is, not opened through a model queue. No security is required to delete a temporary dynamic queue.

## Profiles for processes

If process security is active, you must define profiles in the MQPROC or GMQPROC classes and permit the necessary groups or user IDs access to these profiles, so they can use MQI requests that use processes. Profiles for processes take the form:

```
hlq.processname
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `processname` is the name of the process being opened.

| A profile prefixed by the queue manager name controls access to a single process definition on that queue manager. A profile prefixed by the queue-sharing group name controls access to one or more process definitions with that name on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that process definition on that queue manager.

| If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

The following table shows the access required for opening a process.

*Table 30. Access levels for process security*

| **MQOPEN option** | **RACF access level required to hlq.processname** |
|---|---|
| MQOO_INQUIRE | READ |

For example, on queue manager MQS9, the RACF group INQVPRC must be able to inquire (**MQINQ**) on all processes starting with the letter V. The RACF definitions for this would be:

```
RDEFINE MQPROC MQS9.V* UACC(NONE)
PERMIT MQS9.V* CLASS(MQPROC) ID(INQVPRC) ACCESS(READ)
```

Alternate user security might also be active, depending on the open options specified when a process definition object is opened.

## Profiles for namelists

If namelist security is active, you define profiles in the MQNLIST or GMQNLIST classes and give the necessary groups or user IDs access to these profiles.

Profiles for namelists take the form:

```
hlq.namelistname
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `namelistname` is the name of the namelist being opened.

A profile prefixed by the queue manager name controls access to a single namelist on that queue manager. A profile prefixed by the queue-sharing group name controls access to access to one or more namelists with that name on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that namelist on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

The following table shows the access required for opening a namelist.

*Table 31. Access levels for namelist security*

| **MQOPEN option** | **RACF access level required to hlq.namelistname** |
|---|---|
| MQOO_INQUIRE | READ |

For example, on queue manager (or queue-sharing group) PQM3, the RACF group DEPT571 must be able to inquire (**MQINQ**) on these namelists:
- All namelists starting with “DEPT571”.
- PRINTER/DESTINATIONS/DEPT571
- AGENCY/REQUEST/QUEUES
- WAREHOUSE.BROADCAST

The RACF definitions to do this are:

```
RDEFINE MQNLIST PQM3.DEPT571.** UACC(NONE)
PERMIT PQM3.DEPT571.** CLASS(MQNLIST) ID(DEPT571) ACCESS(READ)

RDEFINE GMQNLIST NLISTS.FOR.DEPT571 UACC(NONE)
        ADDMEM(PQM3.PRINTER/DESTINATIONS/DEPT571,
               PQM3.AGENCY/REQUEST/QUEUES,
               PQM3.WAREHOUSE.BROADCAST)
PERMIT NLISTS.FOR.DEPT571 CLASS(GMQNLIST) ID(DEPT571) ACCESS(READ)
```

Alternate user security might be active, depending on the options specified when a namelist object is opened.

## Profiles for alternate user security

If alternate user security is active, you must define profiles in the MQADMIN class and permit the necessary groups or user IDs access to these profiles, so that they can use the ALTERNATE_USER_AUTHORITY options when the object is opened.

Profiles for alternate user security can be specified at subsystem level or at queue-sharing group level and take the following form:

```
hlq.ALTERNATE.USER.alternateuserid
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `alternateuserid` is the value of the *AlternateUserId* field in the object descriptor.

A profile prefixed by the queue manager name controls use of an alternate user ID on that queue manager. A profile prefixed by the queue-sharing group name controls use of an alternate user ID on all queue managers within the queue-sharing group. This alternate user ID can be used on any queue manager within the queue-sharing group by a user that has the correct access. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that alternate user ID on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

The following table shows the access when specifying an alternate user option.

*Table 32. Access levels for alternate user security*

| MQOPEN or MQPUT1 option | RACF access level required |
|---|---|
| MQOO_ALTERNATE_USER_AUTHORITY<br>MQPMO_ALTERNATE_USER_AUTHORITY | UPDATE |

In addition to alternate user security checks, other security checks for queue, process, namelist, and context security can also be made. The alternate user ID, if provided, is only used for security checks on queue, process definition, or namelist resources. For alternate user and context security checks, the user ID requesting the check is used. For details about how user IDs are handled, see “Chapter 15. User IDs for security checking” on page 181. For a summary table showing the open options and the security checks required when queue, context and alternate user security are all active, see Table 29 on page 157.

An alternate user profile gives the requesting user ID access to resources associated with the user ID specified in the alternate user ID. For example, the payroll server running under user ID PAYSERV on queue manager QMPY processes requests from personnel user IDs, all of which start with PS. To cause the work performed by the payroll server to be carried out under the user ID of the requesting user, alternate user authority is used. The payroll server knows which user ID to specify as the alternate user ID because the requesting programs generate messages using the MQPMO_DEFAULT_CONTEXT put message option. See “Chapter 15. User IDs for security checking” on page 181 for more details about from where alternate user IDs are obtained.

## Profiles for alternate user security

The following example RACF definitions enable the server program to specify alternate user IDs starting with the characters PS:

```
RDEFINE MQADMIN QMPY.ALTERNATE.USER.PS* UACC(NONE)
PERMIT QMPY.ALTERNATE.USER.PS* CLASS(MQADMIN) ID(PAYSERV) ACCESS(UPDATE)
```

**Notes:**

1. The *AlternateUserId* field in the object descriptor is 12 bytes long. All 12 bytes are used in the profile checks, but only the first eight bytes are used as the user ID by MQSeries. If this user ID truncation is not desirable, application programs making the request should translate any alternate user ID over 8 bytes into something more appropriate.
2. If you specify MQOO_ALTERNATE_USER_AUTHORITY or MQPMO_ALTERNATE_USER_AUTHORITY and you do not specify an *AlternateUserId* field in the object descriptor, a user ID of blanks is used. For the purposes of the alternate user security check the user ID used for the *AlternateUserId* qualifier is -BLANK-. For example `RDEF MQADMIN hlq.ALTERNATE.USER.-BLANK-`.

   If the user is allowed to access this profile, all further checks are made with a user ID of blanks. For details of blank user IDs, see “Blank user IDs and UACC levels” on page 190.

The administration of alternate user IDs is easier if you have a naming convention for user IDs that enables you to use generic alternate user profiles. If they do not, you could use the RACF RACVARS feature. For details about using RACVARS, see the *Security Server (RACF) Security Administrator’s Guide*.

When a message is put to a queue that has been opened with alternate user authority and the context of the message has been generated by the queue manager, the MQMD_USER_IDENTIFIER field is set to the alternate user ID.

## Profiles for context security

If context security is active, you must define a profile in the MQADMIN class called:

```
hlq.CONTEXT
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name).

A profile prefixed by the queue manager name allows control for context security on that queue manager. A profile prefixed by the queue-sharing group name allows control for context on all queue managers within the queue-sharing group. This can be overridden on an individual queue manager by defining a queue-manager level profile for context on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

You must give the necessary groups or user IDs access to this profile. The following table shows the access level required, depending on the specification of the context options when the queue is opened.

*Table 33. Access levels for context security*

| **MQOPEN or MQPUT1 option** | **RACF access level required to hlq.CONTEXT** |
|---|---|
| MQPMO_NO_CONTEXT | No context security check |
| MQPMO_DEFAULT_CONTEXT | No context security check |
| MQOO_SAVE_ALL_CONTEXT | No context security check |
| MQOO_PASS_IDENTITY_CONTEXT<br>MQPMO_PASS_IDENTITY_CONTEXT | READ |
| MQOO_PASS_ALL_CONTEXT<br>MQPMO_PASS_ALL_CONTEXT | READ |
| MQOO_SET_IDENTITY_CONTEXT<br>MQPMO_SET_IDENTITY_CONTEXT | UPDATE |
| MQOO_SET_ALL_CONTEXT<br>MQPMO_SET_ALL_CONTEXT | CONTROL |
| MQOO_OUTPUT or MQPUT1 (USAGE(XMITQ)) | CONTROL |
| **Note:** The user IDs used for distributed queuing require CONTROL access to `hlq.CONTEXT` in order to put messages on the destination queue. See “User IDs used by the channel initiator” on page 185 for information about the user IDs used. | |

If you put commands on the system-command input queue, use the default context put message option to associate the correct user ID with the command.

For example, the MQSeries-supplied utility program CSQUTIL can be used to off-load and reload messages in queues. When off-loaded messages are restored to a queue, the CSQUTIL utility uses the MQOO_SET_ALL_CONTEXT option to return the messages to their original state. In addition to the queue security required by this open option, context authority is also required. For example, if this authority is required by the group BACKGRP on queue manager MQS1, this would be defined by:

```
RDEFINE MQADMIN MQS1.CONTEXT UACC(NONE)
PERMIT MQS1.CONTEXT CLASS(MQADMIN) ID(BACKGRP) ACCESS(CONTROL)
```

Depending on the options specified, and the types of security performed, other types of security checks might also occur when the queue is opened. These include queue security (see “Profiles for queue security” on page 150), and alternate user security (see “Profiles for alternate user security” on page 161). For a summary table showing the open options and the security checks required when queue, context and alternate user security are all active, see Table 29 on page 157.

## Profiles for command security

If you want security checking for commands (so you have not defined the command security switch profile hlq.NO.CMD.CHECKS) you must add profiles to the MQCMDS class.

The names of the RACF profiles for command security checking are based on the command names themselves. These profiles take the form:

```
hlq.verb.pkw
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name).

For example, the profile name for the MQSeries command ALTER QLOCAL in subsystem CSQ1 is:

```
CSQ1.ALTER.QLOCAL
```

A profile prefixed by the queue manager name controls the use of the command on that queue manager. A profile prefixed by the queue-sharing group name controls the use of the command on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that command on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

By setting up command profiles at queue manager level, a user can be restricted from issuing commands on a particular queue manager. Alternatively, you can define one profile for a queue-sharing group for each command verb, and all security checks will take place against that profile instead of individual queue managers.

If both subsystem security and queue-sharing group security are active and a local profile is not found, a command security check will be performed to see if the user has access to a queue-sharing group profile.

If you use the CMDSCOPE attribute to route a command to other queue managers in a queue-sharing group, security is checked on each queue manager where the command is executed, but not necessarily on the queue manager where the command is entered.

Table 34 on page 166 shows, for each MQSeries command, the profiles required for command security checking to be carried out, and the corresponding access level for each profile in the MQCMDS class.

## Profiles for command security

*Table 34. Commands, profiles, and their access levels*

| Command | MQCMDS | | MQADMIN | |
|---|---|---|---|---|
| | **Command profile** | **Access level** | **Command resource profile** | **Access level** |
| ALTER CHANNEL | hlq.ALTER.CHANNEL | ALTER | hlq.CHANNEL.channel | ALTER |
| ALTER NAMELIST | hlq.ALTER.NAMELIST | ALTER | hlq.NAMELIST.namelist | ALTER |
| ALTER PROCESS | hlq.ALTER.PROCESS | ALTER | hlq.PROCESS.process | ALTER |
| ALTER QALIAS | hlq.ALTER.QALIAS | ALTER | hlq.QUEUE.queue | ALTER |
| ALTER QLOCAL | hlq.ALTER.QLOCAL | ALTER | hlq.QUEUE.queue | ALTER |
| ALTER QMGR | hlq.ALTER.QMGR | ALTER | No check | |
| ALTER QMODEL | hlq.ALTER.QMODEL | ALTER | hlq.QUEUE.queue | ALTER |
| ALTER QREMOTE | hlq.ALTER.QREMOTE | ALTER | hlq.QUEUE.queue | ALTER |
| ALTER SECURITY | hlq.ALTER.SECURITY | ALTER | No check | |
| ALTER STGCLASS | hlq.ALTER.STGCLASS | ALTER | No check | |
| ALTER TRACE | hlq.ALTER.TRACE | ALTER | No check | |
| ARCHIVE LOG | hlq.ARCHIVE.LOG | CONTROL | No check | |
| CLEAR QLOCAL | hlq.CLEAR.QLOCAL | ALTER | hlq.QUEUE.queue | ALTER |
| DEFINE BUFFPOOL | hlq.DEFINE.BUFFPOOL | ALTER | No check | |
| DEFINE CHANNEL | hlq.DEFINE.CHANNEL | ALTER | hlq.CHANNEL.channel | ALTER |
| DEFINE MAXSMSGS | hlq.DEFINE.MAXSMSGS | ALTER | No check | |
| DEFINE NAMELIST | hlq.DEFINE.NAMELIST | ALTER | hlq.NAMELIST.namelist | ALTER |
| DEFINE PROCESS | hlq.DEFINE.PROCESS | ALTER | hlq.PROCESS.process | ALTER |
| DEFINE PSID | hlq.DEFINE.PSID | ALTER | No check | |
| DEFINE QALIAS | hlq.DEFINE.QALIAS | ALTER | hlq.QUEUE.queue | ALTER |
| DEFINE QLOCAL | hlq.DEFINE.QLOCAL | ALTER | hlq.QUEUE.queue | ALTER |
| DEFINE QMODEL | hlq.DEFINE.QMODEL | ALTER | hlq.QUEUE.queue | ALTER |
| DEFINE QREMOTE | hlq.DEFINE.QREMOTE | ALTER | hlq.QUEUE.queue | ALTER |
| DEFINE STGCLASS | hlq.DEFINE.STGCLASS | ALTER | No check | |
| DELETE CHANNEL | hlq.DELETE.CHANNEL | ALTER | hlq.CHANNEL.channel | ALTER |
| DELETE NAMELIST | hlq.DELETE.NAMELIST | ALTER | hlq.NAMELIST.namelist | ALTER |
| DELETE PROCESS | hlq.DELETE.PROCESS | ALTER | hlq.PROCESS.process | ALTER |
| DELETE QALIAS | hlq.DELETE.QALIAS | ALTER | hlq.QUEUE.queue | ALTER |
| DELETE QLOCAL | hlq.DELETE.QLOCAL | ALTER | hlq.QUEUE.queue | ALTER |
| DELETE QMODEL | hlq.DELETE.QMODEL | ALTER | hlq.QUEUE.queue | ALTER |
| DELETE QREMOTE | hlq.DELETE.QREMOTE | ALTER | hlq.QUEUE.queue | ALTER |
| DELETE STGCLASS | hlq.DELETE.STGCLASS | ALTER | No check | |
| DISPLAY CHANNEL | hlq.DISPLAY.CHANNEL | READ | No check | |
| DISPLAY CHSTATUS | hlq.DISPLAY.CHSTATUS | READ | No check | |
| DISPLAY CLUSQMGR | hlq.DISPLAY.CLUSQMGR | READ | No check | |
| DISPLAY CMDSERV | hlq.DISPLAY.CMDSERV | READ | No check | |
| DISPLAY DQM | hlq.DISPLAY.DQM | READ | No check | |
| DISPLAY GROUP | hlq.DISPLAY.GROUP | READ | No check | |
| DISPLAY LOG | hlq.DISPLAY.LOG | READ | No check | |
| DISPLAY MAXSMSGS | hlq.DISPLAY.MAXSMSGS | READ | No check | |
| DISPLAY NAMELIST | hlq.DISPLAY.NAMELIST | READ | No check | |
| DISPLAY PROCESS | hlq.DISPLAY.PROCESS | READ | No check | |
| DISPLAY QMGR | hlq.DISPLAY.QMGR | READ | No check | |
| DISPLAY QSTATUS | hlq.DISPLAY.QSTATUS | READ | No check | |

*Table 34. Commands, profiles, and their access levels (continued)*

| Command | MQCMDS | | MQADMIN | |
|---|---|---|---|---|
| | **Command profile** | **Access level** | **Command resource profile** | **Access level** |
| DISPLAY QUEUE | hlq.DISPLAY.QUEUE | READ | No check | |
| DISPLAY THREAD | hlq.DISPLAY.THREAD | READ | No check | |
| DISPLAY SECURITY | hlq.DISPLAY.SECURITY | READ | No check | |
| DISPLAY STGCLASS | hlq.DISPLAY.STGCLASS | READ | No check | |
| DISPLAY TRACE | hlq.DISPLAY.TRACE | READ | No check | |
| DISPLAY USAGE | hlq.DISPLAY.USAGE | READ | No check | |
| MOVE QLOCAL | hlq.MOVE.QLOCAL | ALTER | hlq.QUEUE.from-queue hlq.QUEUE.to-queue | ALTER |
| PING CHANNEL | hlq.PING.CHANNEL | CONTROL | hlq.CHANNEL.channel | CONTROL |
| RECOVER BSDS | hlq.RECOVER.BSDS | CONTROL | No check | |
| REFRESH CLUSTER | hlq.REFRESH.CLUSTER | ALTER | No check | |
| REFRESH SECURITY | hlq.REFRESH.SECURITY | ALTER | No check | |
| RESET CHANNEL | hlq.RESET.CHANNEL | CONTROL | hlq.CHANNEL.channel | CONTROL |
| RESET CLUSTER | hlq.RESET.CLUSTER | CONTROL | No check | |
| RESET QSTATS | hlq.RESET.QSTATS | CONTROL | hlq.QUEUE.queue | CONTROL |
| RESET TPIPE | hlq.RESET.TPIPE | CONTROL | No check | |
| RESOLVE CHANNEL | hlq.RESOLVE.CHANNEL | CONTROL | hlq.CHANNEL.channel | CONTROL |
| RESOLVE INDOUBT | hlq.RESOLVE.INDOUBT | CONTROL | No check | |
| RESUME QMGR | hlq.RESUME.QMGR | CONTROL | No check | |
| RVERIFY SECURITY | hlq.RVERIFY.SECURITY | ALTER | No check | |
| SET LOG | hlq.SET.LOG | CONTROL | No check | |
| START CHANNEL | hlq.START.CHANNEL | CONTROL | hlq.CHANNEL.channel | CONTROL |
| START CHINIT | hlq.START.CHINIT | CONTROL | No check | |
| START CMDSERV | hlq.START.CMDSERV | CONTROL | No check | |
| START LISTENER | hlq.START.LISTENER | CONTROL | No check | |
| START TRACE | hlq.START.TRACE | CONTROL | No check | |
| STOP CHANNEL | hlq.STOP.CHANNEL | CONTROL | hlq.CHANNEL.channel | CONTROL |
| STOP CHINIT | hlq.STOP.CHINIT | CONTROL | No check | |
| STOP CMDSERV | hlq.STOP.CMDSERV | CONTROL | No check | |
| STOP LISTENER | hlq.STOP.LISTENER | CONTROL | No check | |
| STOP QMGR | hlq.STOP.QMGR | CONTROL | No check | |
| STOP TRACE | hlq.STOP.TRACE | CONTROL | No check | |
| SUSPEND QMGR | hlq.SUSPEND.QMGR | CONTROL | No check | |

**Notes:**

- MQSeries does not check the authority of the user who issues the START QMGR command. However, you can use RACF facilities to control access to the START xxxxMSTR command that is issued as a result of the START QMGR command. This is done by controlling access to the MVS.START.STC.xxxxMSTR profile in the RACF operator commands (OPERCMDS) class. For details of this, see the *Security Server (RACF) Security Administrator's Guide*. If you use this technique, and an unauthorized user tries to start MQSeries, MQSeries terminates with a reason code of 00F30216.
- The DISPLAY THREAD and DISPLAY USAGE commands might be issued internally by the queue manager; no authority is checked in these cases.

## Profiles for command resource security

If you have not defined the command resource security switch profile, hlq.NO.CMD.RESC.CHECKS, because you want security checking for resources associated with commands, you must add resource profiles to the MQADMIN class for each resource.

Profiles for command resource security checking take the form:

```
hlq.type.resourcename
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name).

A profile prefixed by the queue manager name controls access to the resources associated with commands on that queue manager. A profile prefixed by the queue-sharing group name controls access to the resources associated with commands on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that command resource on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, MQSeries checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

For example, the RACF profile name for command resource security checking against the model queue CREDIT.WORTHY in subsystem CSQ1 is:

```
CSQ1.QUEUE.CREDIT.WORTHY
```

Because the profiles for all types of command resource are held in the MQADMIN class, the “type” part of the profile name is needed in the profile to distinguish between resources of different types that have the same name. The “type” part of the profile name can be CHANNEL, QUEUE, PROCESS, or NAMELIST. For example, a user might be authorized to define hlq.QUEUE.PAYROLL.ONE, but not authorized to define hlq.PROCESS.PAYROLL.ONE

If the resource type is a queue, and the profile is a queue-sharing group level profile, it controls access to one or more local queues within the queue sharing group, or access to a single shared queue from any queue manager in the queue-sharing group.

Table 34 on page 166 shows for each MQSeries command, the profiles you need to enable command resource security checking to be carried out, and the access level that you need for each in the MQADMIN class.

### Command resource security checking for alias queues

When you define an alias queue, command resource security checks are only performed against the name of the alias queue, not against the name of the target queue to which the alias resolves.

Alias queues can resolve to both local and remote queues. If you do not want to permit users access to certain local or remote queues, you must do both of the following:

1. Do not allow the users access to these local and remote queues.
2. Restrict the users from being able to define aliases for these queues. That is, prevent them from being able to issue DEFINE QALIAS and ALTER QALIAS commands.

## Command resource security checking for remote queues

When you define a remote queue, command resource security checks are performed only against the name of the remote queue. No checks are performed against the names of the queues specified in the RNAME or XMITQ attributes in the remote queue object definition. For more information about the attributes of queues, see the *MQSeries MQSC Command Reference* manual.

# Chapter 14. Using the RESLEVEL security profile

You can define a special profile in the MQADMIN class to control the number of user IDs checked for API-resource security. How this RESLEVEL profile affects API-resource security depends on how you are accessing MQSeries.

This chapter discusses the following subjects:

**Important notes about using RESLEVEL:**

1. RESLEVEL is a very powerful option; it can cause the bypassing of all resource security checks, so that they cannot be audited by RACF.
2. You can use the RESAUDIT system parameter to switch RESLEVEL auditing off.
3. Using the RESLEVEL profile means that normal security audit records are not taken. For example, if you put UAUDIT on a user, the access to the hlq.RESLEVEL profile in MQADMIN is not audited.
4. If you use the RACF WARNING option on the hlq.RESLEVEL profile, no RACF warning messages are produced.
5. If you do not have a RESLEVEL profile defined, you must be careful that no other profile in the MQADMIN class matches hlq.RESLEVEL. For example, if you have a profile in MQADMIN called hlq.** and no hlq.RESLEVEL profile, beware of the consequences of the hlq.** profile because it is used for the RESLEVEL check.

   You should define an hlq.RESLEVEL profile and set the UACC to NONE, rather than not have a RESLEVEL profile at all. You should have as few users or groups in the access list as possible. For details about how to audit RESLEVEL access, see "Auditing considerations" on page 197.
6. If you make any changes to the RESLEVEL profile users must disconnect and connect again before the change takes place. (This includes stopping and restarting the channel initiator if the access that the distributed queuing address space user ID has to the RESLEVEL profile is changed.)

## RESLEVEL and Batch/TSO connections

By default, when an MQSeries resource is being accessed through the Batch/TSO adapter, the user must be authorized to access that resource for the particular operation. You can bypass the security check by setting up an appropriate RESLEVEL definition.

Whether the user is checked or not is based on the user ID used at connect time; the TSO user ID or the job user ID.

For example, you can set up RESLEVEL so that when a user you trust accesses certain resources from Batch/TSO, no API-resource security checks are done; but when a user you do not trust tries to access the same resources, security checks are carried out as normal. You should set up RESLEVEL checking to bypass API-resource security checks only when you sufficiently trust the user and the programs run by that user.

The following table shows the checks made for Batch and TSO connections.

*Table 35. Checks made at different RACF access levels for the Batch/TSO adapter*

| RACF access level | Level of checking |
|---|---|
| NONE | Check the job/TSO (or alternate) user ID. |
| READ | Check the job/TSO (or alternate) user ID. |
| UPDATE | Check the job/TSO (or alternate) user ID. |
| CONTROL | No check. |
| ALTER | No check. |

## RESLEVEL and system utilities

The operations and control panels and the CSQUTIL utility are batch applications. You can use RESLEVEL to bypass security checking for the SYSTEM.COMMAND.INPUT and SYSTEM.COMMAND.REPLY.MODEL queues that they use, but **not** for the dynamic queues SYSTEM.CSQOREXX.* and SYSTEM.CSQUTIL.*. Users must be authorized to use these queues as described in “System queue security” on page 156 in addition to any RESLEVEL authorization they are given.

## RESLEVEL and CICS connections

By default, when an API-resource security check is made on a CICS connection, two user IDs are checked to see if access is allowed to the resource.

### User IDs checked

The first user ID checked is that of the CICS address space. This is the user ID on the job card of the CICS job, or the user ID assigned to the CICS started task by the OS/390 STARTED class or the started procedures table. (It is not the CICS DFLTUSER.)

The second user ID checked is the user ID associated with the CICS transaction.

### Completion codes

If one of these user IDs does not have access to the resource, the request fails with a completion code of MQRC_NOT_AUTHORIZED. Both the CICS address space user ID and the user ID of the person running the CICS transaction must have access to the resource at the correct level.

### How RESLEVEL can affect the checks made

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested. The possible checks are:

- Check the CICS address space user ID and the transaction user ID.
- Check the CICS address space user ID only.
- If the transaction is defined to CICS with RESSEC(NO), check the CICS address space user ID only. (The status of the CICS security is NOT checked when taking into consideration the transaction RESSEC setting. For example, if CICS has been started with SEC=NO, but the transaction has been defined with RESSEC(YES), MQSeries still checks both user IDs.)
- If the transaction is defined to CICS with RESSEC(YES), check the CICS address space user ID and the transaction user ID.
- Do not check any user IDs.

The user IDs checked depend on the user ID used at connection time, that is, the CICS address space user ID. This control enables you to bypass API-resource security checking for MQSeries requests coming from one system (for example, a test system, TESTCICS,) but to implement them for another (for example, a production system, PRODCICS).

**Note:** If you set up your CICS address space user ID with the “trusted” attribute in the STARTED class or the RACF started procedures table ICHRIN03, this overrides any user ID checks for the CICS address space established by the RESLEVEL profile for your queue manager (that is, the queue manager does not perform the security checks for the CICS address space). For more information, see the *CICS RACF Security Guide*.

The following table shows the checks made for CICS connections.

*Table 36. Checks made at different RACF access levels for the CICS adapter*

| RACF access level | Level of checking |
|---|---|
| NONE | Check the CICS address space user ID and the task or alternate user ID. |
| READ | Check the CICS address space user ID. |

Table 36. Checks made at different RACF access levels for the CICS adapter (continued)

| RACF access level | Level of checking |
|---|---|
| UPDATE | Check the CICS address space user ID and, if the transaction has been defined with RESSEC=YES, also check the task or alternate user ID. |
| CONTROL | No check. |
| ALTER | No check. |

## RESLEVEL and IMS connections

By default, when an API-resource security check is made for an IMS adapter, two user IDs are checked to see if access is allowed to the resource.

The first user ID checked is that of the address space of the IMS region. This is taken from either the USER field from the job card or the user ID assigned to the region from the OS/390 STARTED class or the started procedures table (SPT).

The second user ID checked is associated with the work being done in the dependent region. It is determined according to the type of the dependent region as shown in Table 43 on page 184.

The setting of MQ RESLEVEL profiles cannot alter the user ID under which IMS transactions are scheduled from the IBM-supplied MQ-IMS trigger monitor program CSQQTRMN. This user ID is the PSBNAME of that trigger monitor, which by default is CSQQTRMN.

### Completion codes

If either the first or second IMS user ID does not have access to the resource, the request fails with a completion code of MQRC_NOT_AUTHORIZED.

### How RESLEVEL can affect the checks made

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested. The possible checks are:

- Check the IMS region address space user ID and the second user ID.
- Check IMS region address space user ID only.
- Do not check any user IDs.

The following table shows the checks made for IMS connections.

*Table 37. Checks made at different RACF access levels for the IMS adapter*

| RACF access level | Level of checking |
|---|---|
| NONE | Check the IMS address space user ID and the IMS second user ID. |
| READ | Check the IMS address space user ID. |
| UPDATE | Check the IMS address space user ID. |
| CONTROL | No check. |
| ALTER | No check. |

## RESLEVEL and channel initiator connections

By default, when an API-resource security check is made by the channel initiator, two user IDs are checked to see if access is allowed to the resource.

The user IDs checked can be that specified by the MCAUSER channel attribute, that received from the network, that of the channel initiator address space, or the alternate user ID for the message descriptor. This depends on the communication protocol you are using and the setting of the PUTAUT channel attribute. See "User IDs used by the channel initiator" on page 185 for more information.

### Completion codes

If one of these user IDs does not have access to the resource, the request fails with a completion code of MQRC_NOT_AUTHORIZED.

### How RESLEVEL can affect the checks made

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested, and how many are checked.

The following table shows the checks made for channel initiator connections.

*Table 38. Checks made at different RACF access levels for channel initiator connections*

| **RACF access level** | **Level of checking** |
|---|---|
| NONE | Check two user IDs. |
| READ | Check one user ID. |
| UPDATE | Check one user ID. |
| CONTROL | No check. |
| ALTER | No check. |
| **Note:** See "User IDs used by the channel initiator" on page 185 for a definition of the user IDs checked | |

| 
## RESLEVEL and intra-group queuing

| By default, when an API-resource security check is made by the intra-group
| queuing agent, two user IDs are checked to see if access is allowed to the resource.

| The user IDs checked can be the user ID determined by the IGQUSER attribute of
| the receiving queue manager, the user ID of the queue manager within the
| queue-sharing group that put the message on to the
| SYSTEM.QSG.TRANSMIT.QUEUE, or the alternate user ID specified in the
| `UserIdentifier` field of the message descriptor of the message. See "User IDs used
| by the intra-group queuing agent" on page 189 for more information.

## The RESLEVEL profile

When an application tries to connect to MQSeries, MQSeries checks the access that the user ID associated with the adapter has to a profile in the MQADMIN class called:

```
hlq.RESLEVEL
```

where `hlq` can be either `ssid` (subsystem ID) or `qsg` (queue-sharing group ID).

The user IDs associated with each adapter are:
- The batch job ID or TSO user ID for the Batch/TSO adapter
- The CICS address space user ID for the CICS adapter
- The IMS region address space user ID for the IMS adapter
- The channel initiator address space user ID for the channel initiator.

If you are using queue manager level security only, MQSeries performs RESLEVEL checks against the `qmgr-name.RESLEVEL` profile. If you are using queue-sharing group level security only, MQSeries performs RESLEVEL checks against the `qsg-name.RESLEVEL` profile. If you are using a combination of both queue manager and queue-sharing group level security, MQSeries first checks for the existence of a RESLEVEL profile at queue manager level. If it does not find one, it checks for a RESLEVEL profile at queue-sharing group level.

If it is unable to find a RESLEVEL profile, MQSeries enables checking of both the job and task (or alternate user) ID for a CICS or an IMS connection. For a batch connection, MQSeries enables checking of the job (or alternate) user ID. For the channel initiator, MQSeries enables checking of the channel user ID and the MCA (or alternate) user ID.

If there is a RESLEVEL profile, the level of checking depends on the environment and access level for the profile.

Remember that if your queue manager is a member of a queue-sharing group and you do not define this profile at queue-manager level, there might be one defined at queue-sharing group level that will effect the level of checking. To activate the checking of two user IDs, you should define a RESLEVEL profile (prefixed with either the queue manager name of the queue-sharing group name) with a UACC(NONE) and ensure that the relevant users do not have access granted against this profile.

### RESLEVEL and the user IDs checked

Table 40 through Table 46 on page 187 show how RESLEVEL affects which user IDs are checked for different MQI requests.

For example, you have a queue manager called QM66, where:
- User WS21B is to be exempt from resource security.
- CICS started task WXNCICS running under address space user ID CICSWXN is to perform full resource checking only for transactions defined with RESSEC(YES).

### RESLEVEL security profile

To define the appropriate RESLEVEL profile, issue the RACF command:

```
RDEFINE MQADMIN QM66.RESLEVEL UACC(NONE)
```

Then give the users access to this profile:

```
PERMIT QM66.RESLEVEL CLASS(MQADMIN) ID(WS21B) ACCESS(CONTROL)
PERMIT QM66.RESLEVEL CLASS(MQADMIN) ID(CICSWXN) ACCESS(UPDATE)
```

If you make these changes while the user IDs are connected to queue manager QM66, the users must disconnect and connect again before the change takes place.

If subsystem security is not active when a user connects but, while this user is still connected, subsystem security becomes active, full resource security checking is applied to the user. The user must re-connect to get the correct RESLEVEL processing.

## The RESLEVEL profile and intra-group queuing

Because the intra-group queuing agent is an internal queue manager task, it does not issue an explicit connect request and runs under the user ID of the queue manager. The intra-group queuing agent starts at queue manager initialization. During the initialization of the intra-group queuing agent, MQSeries checks the access that the user ID associated with the queue manager has to a profile in the MQADMIN class called:

```
hlq.RESLEVEL
```

This check is always performed unless the ssid.NO.SUBSYS.SECURITY switch has been set.

If there is no RESLEVEL profile, MQSeries enables checking for two user IDs. If there is a RESLEVEL profile, the level of checking depends on the access level granted to the user ID of the queue manager for the profile. Table 39 shows the checks made for the intra-group queuing agent.

*Table 39. Checks made at different RACF access levels for the intra-group queuing agent*

| **RACF access level** | **Level of checking** |
|---|---|
| NONE | Check two user IDs. |
| READ | Check one user ID. |
| UPDATE | Check one user ID. |
| CONTROL | No check. |
| ALTER | No check. |
| **Note:** See "User IDs used by the intra-group queuing agent" on page 189 for a definition of the user IDs checked | |

If the permissions granted to the RESLEVEL profile for the queue manager's user ID are changed, the intra-group queuing agent must be stopped and restarted to pick up the new permissions. Because there is no way to independently stop and restart the intra-group queuing agent, the queue manager must be stopped and restarted to achieve this.

# Chapter 15. User IDs for security checking

MQSeries initiates security checks based on user IDs associated with users, terminals, applications, and so on. The following sections show the contents of the user IDs used for each type of security check.

This chapter discusses the following topics:
- "User IDs for connection security"
- "User IDs for command security and command resource security"
- "User IDs for resource security (MQOPEN and MQPUT1)" on page 182
- "Blank user IDs and UACC levels" on page 190

## User IDs for connection security

| **Issued from...** | **User ID contents** |
|---|---|
| Batch/TSO, CICS, IMS | The user ID normally found for connection security is one of these:<br>• The TSO user ID.<br>• The user ID assigned to a batch job via the USER JCL parameter.<br>• The user ID assigned to a started task by the STARTED class or the started procedures table. |

## User IDs for command security and command resource security

| **Issued from...** | **User ID contents** |
|---|---|
| CSQINP1 or CSQINP2 | No check is made. |
| System command input queue | The user ID found in the *UserIdentifier* of the message descriptor of the message which contains the command. If the message does not contain a *UserIdentifier*, a user ID of blanks is passed to the security manager. |
| Console | The user ID signed onto the console. If the console is not signed on, the default user ID set by the CMDUSER system parameter in CSQ6SYSP.<br><br>To issue commands from a console, the console must have the OS/390 SYS AUTHORITY attribute. |
| SDSF/TSO console | TSO or job user ID. |
| Operations and control panels | TSO user ID.<br><br>If you are going to use the operations and control panels, you must have the appropriate authority to issue the commands corresponding to the actions that you choose. In addition, you must have READ access to all the hlq.DISPLAY.*object* profiles in the MQCMDS class because the panels use the various DISPLAY commands to gather the information that they present. |
| MGCR (SVC34) or MGCRE | If MGCR is used with Utoken, the user ID in the Utoken.<br><br>If MGCR is issued without the Utoken, the TSO or job user ID is used. |
| CSQUTIL | Job user ID. |
| CSQINPX | User ID of the channel initiator address space. |

## User IDs for resource security (MQOPEN and MQPUT1)

Table 40 through Table 46 on page 187 show the contents of the user IDs for normal and alternate user IDs for each type of adapter. The number of checks is defined by the RESLEVEL profile. The user ID checked is that used for **MQOPEN** or **MQPUT1** calls.

**Note:** All user ID fields are checked *exactly* as they are received. No conversions take place, and, for example, three user ID fields containing Bob, BOB, and bob are not equivalent.

### User IDs checked for Batch

*Table 40. User ID checking for Batch/TSO-type user IDs*

| **Profile name** | **Alternate user ID specified on open?** | |
|---|---|---|
| | **No** | **Yes** |
| hlq.ALTERNATE.USER.id | – | JOB |
| hlq.CONTEXT | JOB | JOB |
| hlq.resourcename | JOB | ALT |

Key:

**ALT** Alternate user ID.

**JOB**
- The TSO user ID.
- The user ID assigned to a batch job.
- The user ID assigned to a started task by the STARTED class or the started procedures table.

#### Batch example

A Batch job is performing an **MQPUT1** to a queue called Q1 with RESLEVEL set to READ and alternate user ID checking turned off.

Table 35 on page 172 and Table 40 show that the job user ID is checked against profile hlq.Q1.

## User IDs checked for CICS

Table 41. User ID checking for CICS-type user IDs

| Profile name | Alternate user ID specified on open? | | | |
|---|---|---|---|---|
| | No | | Yes | |
| | 1 Check | 2 Checks | 1 Check | 2 Checks |
| hlq.ALTERNATE.USER.userid | – | – | ADS | ADS+TXN |
| hlq.CONTEXT | ADS | ADS+TXN | ADS | ADS+TXN |
| hlq.resourcename | ADS | ADS+TXN | ADS | ADS+ALT |

Key:

| | |
|---|---|
| **ALT** | Alternate user ID |
| **ADS** | The user ID associated with the CICS batch job or, if CICS is running as a started task, through the STARTED class or the started procedures table. |
| **TXN** | The user ID associated with the CICS transaction. This is normally the user ID of the terminal user who started the transaction. It can be the CICS DFLTUSER, a PRESET security terminal, or a manually signed-on user. |

### CICS example

Determine the user IDs checked for the following conditions:

- The RACF access level to the RESLEVEL profile, for a CICS address space user ID, is set to NONE.
- An **MQOPEN** call is made against a queue with MQOO_OUTPUT and MQOO_PASS_IDENTITY_CONTEXT.

***Answer:*** First, see how many CICS user IDs are checked based on the CICS address space user ID access to the RESLEVEL profile. From Table 36 on page 173, two user IDs are checked if the RESLEVEL profile is set to NONE. Then, from Table 41, these checks are carried out:

- The hlq.ALTERNATE.USER.userid profile is not checked.
- The hlq.CONTEXT profile is checked with both the CICS address space user ID and the CICS transaction user ID.
- The hlq.resourcename profile is checked with both the CICS address space user ID and the CICS transaction user ID.

This means that four security checks are made for this **MQOPEN** call.

## User IDs checked for IMS

*Table 42. User ID checking for IMS-type user IDs*

| **Profile name** | **Alternate user ID specified on open?** | | | |
|---|---|---|---|---|
| | **No** | | **Yes** | |
| | **1 Check** | **2 Checks** | **1 Check** | **2 Checks** |
| hlq.ALTERNATE.USER.id | – | – | REG | REG+SEC |
| hlq.CONTEXT | REG | REG+SEC | REG | REG+SEC |
| hlq.resourcename | REG | REG+SEC | REG | REG+ALT |

Key:

**ALT** Alternate user ID.

**REG** The user ID is normally set through the STARTED class or the started procedures table or, if IMS is running, from a submitted job, via the USER JCL parameter.

**SEC** The second user ID is associated with the work being done in a dependent region. It is determined according to Table 43.

*Table 43. How the second user ID is determined for the IMS adapter*

| **Types of dependent region** | **Hierarchy for determining the second user ID** |
|---|---|
| • BMP message driven and successful GET UNIQUE issued.<br>• IFP and GET UNIQUE issued.<br>• MPP. | User ID associated with the IMS transaction if the user is signed on.<br><br>LTERM name if available.<br><br>PSBNAME. |
| • BMP message driven and successful GET UNIQUE not issued.<br>• BMP not message driven.<br>• IFP and GET UNIQUE not issued. | User ID associated with the IMS dependent region address space if this is not all blanks or all zeros.<br><br>PSBNAME. |

## User IDs used by the channel initiator

The following sections describe the user IDs used and checked for the following:

- TCP/IP receiving channels.
- LU 6.2 receiving channels.
- Client MQI requests issued over server-connection channels for both TCP/IP and LU 6.2.

You can use the PUTAUT parameter of the receiving channel definition to determine the type of security checking used. To get consistent security checking throughout your MQSeries network, you can use the ONLYMCA and ALTMCA options.

### Receiving channels using TCP/IP

**MCA user ID (MCA)**
The user ID specified for the MCAUSER channel attribute at the receiver; if blank, the channel initiator address space user ID of the receiver or requester side is used.

**Channel user ID (CHL)**
On TCP/IP, security is not supported by the communication system for the channel. Because of this, the user ID of the channel initiator address space of the receiver or requestor end is used as the channel user ID on channels defined with the PUTAUT parameter set to DEF or CTX.

If the PUTAUT parameter is set to ONLYMCA or ALTMCA for the channel, the channel user ID is ignored and the MCA user ID of the receiver or requester is used.

**Alternate user ID (ALT)**
The user ID from the context information (that is, the *UserIdentifier* field) within the message descriptor of the message. This user ID is moved into the *AlternateUserID* field in the object descriptor before an **MQOPEN**or **MQPUT1** call is issued for the target destination queue.

Table 44. User IDs checked for TCP/IP channels

| Profile Name | PUTAUT option specified on receiver or requester channel | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | DEF | | CTX | | ONLYMCA | | ALTMCA | |
| | 1 Check | 2 Checks | 1 Check | 2 Checks | 1 Check | 2 Checks | 1 Check | 2 Checks |
| hlq.ALTERNATE.USER.id | - | - | CHL | CHL + MCA | - | - | MCA | MCA |
| hlq.CONTEXT | CHL | CHL + MCA | CHL | CHL + MCA | MCA | MCA | MCA | MCA |
| hlq.resourcename | CHL | CHL + MCA | CHL | CHL + ALT | MCA | MCA | MCA | MCA + ALT |

Key:

| | |
|---|---|
| **ALT** | Alternate user ID. |
| **CHL** | Channel user ID. |
| **MCA** | MCA user ID. |

### Receiving channels using LU 6.2

**MCA user ID (MCA)**
The user ID specified for the MCAUSER channel attribute at the receiver; if blank, the channel initiator address space user ID of the receiver or requester side is used.

**Channel user ID (CHL)**

**Requester-server channels**
If the channel is started from the requester, there is no opportunity to receive a network user ID (the channel user ID).

If the PUTAUT parameter is set to DEF or CTX on the requester channel, the channel user ID is that of the channel initiator address space of the requester because no user ID is received from the network.

If the PUTAUT parameter is set to ONLYMCA or ALTMCA, the channel user ID is ignored and the MCA user ID of the requester is used.

**Other channel types**
If the PUTAUT parameter is set to DEF or CTX on the receiver or requester channel, the channel user ID is the user ID received from the communications system when the channel is initiated.

- If the sending channel is on OS/390, the channel user ID received is the channel initiator address space user ID of the sender.
- If the sending channel is on a different platform (for example, AIX® or HP-UX), the channel user ID received is typically provided by the USERID parameter of the channel definition.

If the user ID received is blank, or no user ID is received, a channel user ID of blanks is used.

**Alternate user ID (ALT)**
The user ID from the context information (that is, the *UserIdentifier* field) within the message descriptor of the message. This user ID is moved into the *AlternateUserID* field in the object descriptor before an **MQOPEN**or **MQPUT1** call is issued for the target destination queue.

*Table 45. User IDs checked for LU 6.2 channels*

| Profile Name | PUTAUT option specified on receiver or requester channel | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | DEF | | CTX | | ONLYMCA | | ALTMCA | |
| | 1 Check | 2 Checks | 1 Check | 2 Checks | 1 Check | 2 Checks | 1 Check | 2 Checks |
| hlq.ALTERNATE.USER.id | - | - | CHL | CHL + MCA | - | - | MCA | MCA |
| hlq.CONTEXT | CHL | CHL + MCA | CHL | CHL + MCA | MCA | MCA | MCA | MCA |
| hlq.resourcename | CHL | CHL + MCA | CHL | CHL + ALT | MCA | MCA | MCA | MCA + ALT |

Key:

| | |
|---|---|
| **ALT** | Alternate user ID. |
| **CHL** | Channel user ID. |
| **MCA** | MCA user ID. |

## Client MQI requests

This section describes the user IDs checked for client MQI requests issued over server-connection channels for TCP/IP and LU 6.2. The MCA user ID and channel user ID are as for the TCP/IP and LU 6.2 channels described in the previous sections.

For server-connection channels, the MCA user ID received from the client is used if the MCAUSER attribute is blank. However, for the clients that can use the MQ_USER_ID environment variable to supply the user ID, it is possible that no environment variable has been set. In this case, the user ID that started the server channel is used. This is the user ID assigned to the channel initiator started task by the OS/390 started procedures table.

See the *MQSeries Clients* manual for more information.

For client **MQOPEN** and **MQPUT1** requests, use the following rules to determine the profile that will be checked:

- If the request specifies alternate-user authority, a check is made against the *hlq*.ALTERNATE.USER.*userid* profile.
- If the request specifies context authority, a check is made against the *hlq*.CONTEXT profile.
- For all **MQOPEN** and **MQPUT1** requests, a check is made against the *hlq.resourcename* profile.

When you have determined which profiles are checked, use the following table to determine which user IDs are checked against these profiles.

*Table 46. User IDs checked for LU 6.2 and TCP/IP server-connection channels*

| **Profile Name** | **PUTAUT option specified on server-connection channel** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **DEF** | | | | **ONLYMCA** | | | |
| | **Alternate user ID specified on open?** | | | | **Alternate user ID specified on open?** | | | |
| | **No** | | **Yes** | | **No** | | **Yes** | |
| | **1 Check** | **2 Checks** | **1 Check** | **2 Checks** | **1 Check** | **2 Checks** | **1 Check** | **2 Checks** |
| hlq.ALTERNATE.USER.id | - | - | CHL | CHL + MCA | - | - | MCA | MCA |
| hlq.CONTEXT | CHL | CHL + MCA | CHL | CHL + MCA | MCA | MCA | MCA | MCA |
| hlq.resourcename | CHL | CHL + MCA | CHL | CHL + ALT | MCA | MCA | MCA | MCA + ALT |
| Key:<br>**ALT** Alternate user ID.<br>**CHL** Channel user ID.<br>**MCA** MCA user ID. | | | | | | | | |

## Channel initiator example

A user performs an **MQPUT1** operation to a queue on queue manager QM01 that resolves to a queue called QB on queue manager QM02. The message is sent on a TCP/IP channel called QM01.TO.QM02. RESLEVEL is set to NONE, and the open will be performed with alternate user ID and context checking. The receiver channel definition has PUTAUT(CTX) and the MCA user ID is set. Which user IDs will be used on the receiving channel to put the message to queue QB?

***Answer:*** Table 38 on page 176 shows that two user IDs are checked because RESLEVEL is set to NONE.

Table 44 on page 185 shows that, with PUTAUT set to CTX and 2 checks, the following user IDs are checked:

- The channel initiator user ID and the MCAUSER user ID are checked against the hlq.ALTERNATE.USER.userid profile.
- The channel initiator user ID and the MCAUSER user ID are checked against the hlq.CONTEXT profile.
- The channel initiator user ID and the alternate user ID specified in the message descriptor (MQMD) are checked against the hlq.Q2 profile.

## User IDs used by the intra-group queuing agent

This section describes the user IDs that are checked when the intra-group queuing agent opens destination queues. The user IDs used are determined by the values of the IGQAUT and IGQUSER queue manager attributes. The possible user IDs are:

**Intra-group queuing user ID (IGQ)**
The user ID determined by the IGQUSER attribute of the receiving queue manager. If this is set to blanks, the user ID of the receiving queue manager is used. However, because the receiving queue manager has authority to access all queues defined to it, security checks are not performed for the receiving queue manager's user ID. In this case:

- If only one user ID is to be checked and the user ID is that of the receiving queue manager, no security checks will take place. This can occur when IGAUT is set to ONLYIGQ or ALTIGQ.
- If two user IDs are to be checked and one of the user IDs is that of the receiving queue manager, security checks will take place for the other user ID only. This can occur when IGAUT is set to DEF, CTX, or ALTIGQ.
- If two user IDs are to be checked and both user IDs are that of the receiving queue manager, no security checks will take place. This can occur when IGAUT is set to ONLYIGQ.

**Sending queue manager user ID (SND)**
The user ID of the queue manager within the queue-sharing group that put the message on to the SYSTEM.QSG.TRANSMIT.QUEUE.

**Alternate user ID (ALT)**
The user ID specified in the *UserIdentifier* field in the message descriptor of the message.

*Table 47. User IDs checked for intra-group queuing*

| Profile Name | IGQAUT option specified on receiving queue manager | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | DEF | | CTX | | ONLYIGQ | | ALTIGQ | |
| | 1 Check | 2 Checks | 1 Check | 2 Checks | 1 Check | 2 Checks | 1 Check | 2 Checks |
| hlq.ALTERNATE.USER.id | - | - | SND | SND + IGQ | - | - | IGQ | IGQ |
| hlq.CONTEXT | SND | SND +IGQ | SND | SND +IGQ | IGQ | IGQ | IGQ | IGQ |
| hlq.resourcename | SND | SND +IGQ | SND | SND + ALT | IGQ | IGQ | IGQ | IGQ + ALT |

Key:

| | |
|---|---|
| **ALT** | Alternate user ID. |
| **IGQ** | IGQ user ID. |
| **SND** | Sending queue manager user ID. |

## Blank user IDs and UACC levels

Blank user IDs can exist when a user is manipulating messages using context or alternate-user security, or when MQSeries is passed a blank user ID. For example, a blank user ID is used when a message is written to the system-command input queue without context.

**Note:** A user ID of '* ' (that is, an asterisk character followed by seven spaces) is treated as a blank user ID.

MQSeries passes the blank user ID to RACF and a RACF undefined user is signed on. All security checks then use the universal access (UACC) for the relevant profile. Depending on how you have set your access levels, the UACC might give the undefined user a wide-ranging access.

For example, if you issue this RACF command from TSO:

```
RDEFINE MQQUEUE Q.AVAILABLE.TO.EVERYONE UACC(UPDATE)
```

you define a profile that enables both OS/390-defined user IDs (that have not been put in the access list) and the RACF undefined user ID to put messages on, and get messages from, that queue.

To protect your MQSeries subsystem from blank user IDs you must plan your access levels carefully, and limit the number of people who can use context and alternate-user security. You must prevent people using the RACF undefined user ID from getting access to resources that they should not. However, at the same time, you must allow access to people with defined user IDs. To do this, you can specify a user ID of asterisk (*) in a RACF command PERMIT. For example, these RACF commands prevent the RACF undefined user ID from gaining access to the queue to put or get messages:

```
RDEFINE MQQUEUE Q.AVAILABLE.TO.RACF.DEFINED.USERS.ONLY UACC(NONE)
PERMIT Q.AVAILABLE.TO.RACF.DEFINED.USERS.ONLY CLASS(MQQUEUE) ACCESS(UPDATE) ID(*)
```

# Chapter 16. MQSeries security management

MQSeries uses an in-storage table to hold information relating to each user and the access requests made by each user.

To manage this table efficiently and to reduce the number of requests made from MQSeries to the external security manager (ESM), these controls are available:
- User ID reverification
- User ID timeouts
- Security refreshes
- Displaying security status

These controls are available through both the operations and control panels and MQSC.

The chapter also discusses the following subjects:
- “Security installation tasks” on page 195
- “Auditing considerations” on page 197
- “Customizing security” on page 198
- “Security problem determination” on page 199

## User ID reverification

If the RACF definition of a user who is using MQSeries resources has been changed—for example, by connecting the user to a new group—you can tell the queue manager to sign this user on again the next time it tries to access an MQSeries resource. You can do this by using the MQSeries command RVERIFY SECURITY. For example:

- User HX0804 is getting and putting messages to the PAYROLL queues on queue manager PRD1. However HX0804 now requires access to some of the PENSION queues on the same queue manager (PRD1).
- The data security administrator connects user HX0804 to the RACF group that allows access to the PENSION queues.
- So that HX0804 can access the PENSION queues immediately—that is, without shutting down queue manager PRD1, or waiting for HX0804 to time out—you must use the MQSeries command:

```
RVERIFY SECURITY(HX0804)
```

**Note:** If you turn off user ID timeout for long periods of time (days or even weeks), while the queue manager is running, you must remember to perform an RVERIFY SECURITY for any users that have been revoked or deleted in that time.

## User ID timeouts

When a user accesses an MQSeries resource, the queue manager tries to sign this user on to the queue manager (if subsystem security is active). This means that the user is authenticated to the ESM. This user remains signed on to MQSeries until either the queue manager is shut down, or until the user ID is “timed out” (the authentication lapses) or reverified (reauthenticated).

When a user is timed out, the user ID is “signed off” within the queue manager and any security-related information retained for this user is discarded. The signing on and off of the user within the queue manager is transparent to the application program and to the end user.

Users are eligible for time out when they have not used any MQSeries resources for a predetermined amount of time. This time period is set by the MQSeries ALTER SECURITY command. For a description of the command syntax, see the *MQSeries MQSC Command Reference* manual.

Two values can be specified in the ALTER SECURITY command:

**TIMEOUT**
The time period in minutes that an unused user ID can remain signed on within the MQSeries subsystem.

**INTERVAL**
The time period in minutes between MQSeries checks for user IDs for which the TIMEOUT has expired.

For example, if the TIMEOUT value is 30 and the INTERVAL value is 10, every 10 minutes MQSeries checks for user IDs that have not been used for 30 minutes. If such a user ID is found, that user ID is signed off within the queue manager. If you do not want to time out user IDs, set the INTERVAL value to zero.

Tuning this value can be important if you have many one-off users. If you set small interval and timeout values, the system is cleaned up more frequently and storage that is no longer required is freed for reuse.

**Note:** If you use values for INTERVAL or TIMEOUT other than the defaults, you must re-enter the command at every MQSeries startup. You can do this automatically by putting the MQSeries command ALTER SECURITY in the CSQINP1 data set for that queue manager.

## Security refreshes

Whenever you add, change or delete a RACF resource profile that is held in the MQADMIN, MQPROC, MQQUEUE, or MQNLIST class, you must tell the queue managers that use this class to refresh the security information that they hold. To do this, issue the following two commands:

1. The RACF SETROPTS RACLIST(classname) REFRESH command to refresh at the RACF level.
2. The MQSeries REFRESH SECURITY command to refresh the security information held by the queue manager (described in the *MQSeries MQSC Command Reference* manual). This command needs to be issued by each queue manager that accesses the profiles that have changed. If you have a queue-sharing group, you can use the command scope attribute to direct the command to all the queue managers in the group.

If you are using generic profiles in any of the MQSeries classes, you must also issue normal RACF refresh commands if you change, add, or delete any generic profiles. For example, SETROPTS GENERIC(classname) REFRESH.

If you change your security settings by adding or deleting switch profiles in the MQADMIN class, you need to use the REFRESH SECURITY(*) or REFRESH SECURITY(MQADMIN) command to pick up these changes dynamically. This means you can activate new security types, or de-activate them without having to restart the queue manager. You do not need to issue the SETROPTS REFRESH command for these changes.

For performance reasons, these are the only classes affected by the MQSeries REFRESH SECURITY command. You do *not* need to use REFRESH SECURITY if you change a profile in either the MQCONN or MQCMDS classes.

**Note:** A refresh of MQADMIN is not required if you change a RESLEVEL security profile.

For performance reasons, use REFRESH SECURITY as infrequently as possible, ideally at off-peak times. You can minimize the number of security refreshes by connecting users to RACF groups that are already in the access list for MQSeries profiles, rather than putting individual users in the access lists. In this way, you change the user rather than the resource profile. You can also RVERIFY SECURITY the appropriate user instead of refreshing security.

As an example of REFRESH SECURITY, suppose you define the new profiles to protect access to queues starting with INSURANCE.LIFE on queue manager PRMQ. You use these RACF commands:

```
RDEFINE MQQUEUE PRMQ.INSURANCE.LIFE.** UACC(NONE)
PERMIT PRMQ.INSURANCE.LIFE.** ID(LIFEGRP) ACCESS(UPDATE)
```

You must issue the following command to tell RACF to refresh the security information that it holds, for example:

```
SETROPTS RACLIST(MQQUEUE) REFRESH
```

Because these profiles are generic, you must tell RACF to refresh the generic profiles for MQQUEUE. For example:

```
SETROPTS GENERIC(MQQUEUE) REFRESH
```

Then you must use this command to tell queue manager PRMQ that the queue profiles have changed:

```
REFRESH SECURITY(MQQUEUE)
```

## Displaying security status

To display the status of the security switches, and other security controls, you can issue the MQSeries command DISPLAY SECURITY. For a description of the command syntax, see the *MQSeries MQSC Command Reference* manual.

Figure 28 shows a typical output of the MQSeries command DISPLAY SECURITY ALL.

```
CSQH015I +CSQ1 Security timeout = 54 MINUTES
CSQH016I +CSQ1 Security interval = 12 MINUTES
CSQH030I +CSQ1 Security switches ...
CSQH034I +CSQ1 SUBSYSTEM: ON, 'SQ05.NO.SUBSYS.SECURITY' not found
CSQH032I +CSQ1 QMGR: ON, 'CSQ1.YES.QMGR.CHECKS' found
CSQH031I +CSQ1 QSG: OFF, 'SQ05.NO.QSG.CHECKS' found
CSQH031I +CSQ1 CONNECTION: OFF, 'CSQ1.NO.CONNECT.CHECKS' found
CSQH034I +CSQ1 COMMAND: ON, 'CSQ1.NO.COMMAND.CHECKS' not found
CSQH031I +CSQ1 CONTEXT: OFF, 'CSQ1.NO.CONTEXT.CHECKS' found
CSQH034I +CSQ1 ALTERNATE USER: ON, 'CSQ1.NO.ALTERNATE.USER.CHECKS' not found
CSQH034I +CSQ1 PROCESS: ON, 'CSQ1.NO.PROCESS.CHECKS' not found
CSQH034I +CSQ1 NAMELIST: ON, 'CSQ1.NO.NLIST.CHECKS' not found
CSQH034I +CSQ1 QUEUE: ON, 'CSQ1.NO.QUEUE.CHECKS' not found
CSQH031I +CSQ1 COMMAND RESOURCES: OFF, 'CSQ1.NO.CMD.RESC.CHECKS' found
CSQ9022I +CSQ1 CSQHPDTC ' DISPLAY SECURITY' NORMAL COMPLETION
```

*Figure 28. Typical output from the MQSeries command DISPLAY SECURITY*

The example shows that the queue manager that replied to the command has all MQSeries security active, except for namelist security. It also shows that user ID timeouts are active, and that every 12 minutes the queue manager checks for user IDs that have not been used in this queue manager for 54 minutes and removes them.

**Note:** This command shows the current security status. It does not necessarily reflect the current status of the switch profiles defined to RACF, or the status of the RACF classes. For example, the switch profiles might have been changed since the last restart of this queue manager or REFRESH SECURITY command.

## Security installation tasks

When MQSeries is first installed, you must perform these security-related tasks:

1. Set up MQSeries data set and system security by:
   - Authorizing the queue manager started-task procedure xxxxMSTR and the distributed queuing started-task procedure xxxxCHIN to run under RACF.
   - Authorizing access to queue manager data sets.
   - Authorizing access for those user IDs that will use the queue manager and utility programs.
   - Authorizing access for those queue managers that will use the coupling facility list structures.
   - Authorizing access for those queue managers that will use DB2.
2. Set up RACF definitions for MQSeries security.

## Setting up MQSeries data set and system security

The possible users of MQSeries data sets include:

- The queue manager itself.
- The channel initiator
- MQSeries administrators who need to create MQSeries data sets, run utility programs, and so on.
- Application programmers, who need to use the MQSeries-supplied copybooks, include data sets, macros, and so on.
- Applications involving one or more of the following:
  – Batch jobs
  – TSO users
  – CICS regions
  – IMS regions

For all of these potential users, protect the MQSeries data sets with RACF.

You must also control access to all your ‘CSQINP’ data sets.

### RACF authorization of started-task procedures

Some MQSeries data sets should be for the exclusive use of the queue manager. If you protect your MQSeries data sets using RACF, you must also authorize the queue manager started-task procedure xxxxMSTR, and the distributed queuing started-task procedure xxxxCHIN, using RACF. To do this, use either:

- The STARTED class.
- The started procedures table (ICHRIN03).

  (Any changes you make to the RACF started procedures table require that you IPL your OS/390 system before the changes can take effect.)

For more information, see the *Security Server (RACF) System Programmer’s Guide*.

The RACF user ID identified must have the required access to the data sets in the started-task procedure. For example, if you associate a queue manager started task procedure called CSQ1MSTR with the RACF user ID QMGRCSQ1, the user ID QMGRCSQ1 must have access to the OS/390 resources accessed by the CSQ1 queue manager.

The RACF user IDs associated with the queue manager and channel initiator started task procedures should not have the TRUSTED attribute set.

## Authorizing access to data sets

The MQSeries data sets should be protected so that no unauthorized user can run a queue manager instance, or gain access to any queue manager data. To do this, use normal OS/390 RACF data set protection. For more information, see the *Security Server (RACF) Security Administrator's Guide*.

Table 48 summarizes the RACF access that the queue manager started task procedure must have to the different data sets.

*Table 48. RACF access to data sets associated with a queue manager*

| **RACF access** | **Data sets** |
|---|---|
| READ | • thlqual.SCSQAUTH and thlqual.SCSQANLx (where x is the language letter for your national language).<br>• The data sets referred to by CSQINP1, CSQINP2 and CSQXLIB in the queue manager's started task procedure. |
| UPDATE | • All page sets and log and BSDS data sets. |
| ALTER | • All archive data sets. |

Table 49 summarizes the RACF access that the started task procedure for distributed queuing must have to the different data sets.

*Table 49. RACF access to data sets associated with distributed queuing*

| **RACF access** | **Data sets** |
|---|---|
| READ | • thlqual.SCSQAUTH, thlqual.SCSQANLx (where x is the language letter for your national language), and thlqual.SCSQMVR1 or thlqual.SCSQMVR2.<br>• LE/370 library data sets.<br>• The data sets referred to by CSQXLIB and CSQINPX in the distributed queuing started task procedure. |
| UPDATE | • Data sets CSQOUTX and CSQSNAP<br>• Dynamic queues SYSTEM.CSQXCMD.* |

## Auditing considerations

The normal RACF auditing controls are available for conducting a security audit of a queue manager. The RACF auditing can be based upon:

- User IDs
- Resource classes
- Profiles

For more details, see the *Security Server (RACF) Auditor’s Guide*.

**Note:** Auditing degrades performance; the more auditing you implement, the more performance is degraded. This is also a consideration for the use of the RACF WARNING option.

### Auditing RESLEVEL

You can decide whether to produce RESLEVEL audit records by setting the RESAUDIT system parameter to YES or NO. If the RESAUDIT parameter is set to NO, audit records are not produced. For more details about setting this parameter, see “Using CSQ6SYSP” on page 31.

If RESAUDIT is set to YES, no normal RACF audit records are taken when the RESLEVEL check is made to see what access an address space user ID has to the hlq.RESLEVEL profile. Instead, MQSeries requests that RACF create a GENERAL audit record (event number 27). These checks are only carried out at connect time, so the overhead should be minimal.

You can report the MQSeries general audit records using the RACF report writer (RACFRW). You could use the following RACFRW commands to report the RESLEVEL access:

```
RACFRW
SELECT PROCESS
EVENT GENERAL
LIST
END
```

A sample report from RACFRW, excluding the *Date*, *Time*, and *SYSID* fields, is shown in Figure 29.

```
        RACF REPORT - LISTING OF PROCESS RECORDS                                    PAGE   4
                                E
                                   V  Q
                                   E  U
 *JOB/USER  *STEP/   --TERMINAL--  N  A
    NAME     GROUP      ID    LVL  T  L

  WS21B     MQMGRP  IGJZM000   0   27  0  JOBID=(WS21B 00.111 09:44:57),USERDATA=()
     TRUSTED USER                         AUTH=(NONE),REASON=(NONE)
                                    SESSION=TSOLOGON,TERMINAL=IGJZM000,
                                    LOGSTR='CSQH RESLEVEL CHECK PERFORMED AGAINST PROFILE(QM66.RESLEVEL),
                                    CLASS(MQADMIN), ACCESS EQUATES TO (CONTROL)',RESULT=SUCCESS,MQADMIN
```

*Figure 29. Sample output from RACFRW showing RESLEVEL general audit records*

From checking the LOGSTR data in the output above, you can see that TSO user WS21B has CONTROL access to QM66.RESLEVEL. This means that all resource security checks will be bypassed when user WS21B access QM66 resources.

For more information about using RACFRW, see the *Security Server (RACF) Auditor's Guide*.

### Statistics

MQSeries does not gather any security statistics of its own. The only statistics are those that can be created by auditing.

## Customizing security

If you want to change the way MQSeries security operates, you must do this via the SAF exit (ICHRFR00), or exits in your external security manager. To find out more about RACF exits, see the *Security Server (RACF) External Security Interface (RACROUTE) Macro Reference* manual.

**Note:** Because MQSeries optimizes calls to the ESM, RACROUTE requests might not be made on, for example, every open for a particular queue by a particular user.

## Security problem determination

This section describes the conditions under which violation messages can be generated in an MQSeries application program and provides a checklist to be implemented if the ESM is not controlling access in the way that you expect.

### Violation messages

A return code of MQRC_NOT_AUTHORIZED can be returned to an application program because:

- A user is not allowed to connect to the queue manager. In this case, you get an ICH408I message in the Batch/TSO, CICS, or IMS job log.
- A user signon to the queue manager has failed because, for example, the job user ID is not valid or appropriate, or the task user ID or alternate user ID is not valid. One or more of these user IDs might not be valid because they have been revoked or deleted. In this case, you get an ICHxxxx message and possibly an IRRxxxx message in the queue manager job log giving the reason for the signon failure. For example:

```
ICH408I USER(NOTDFND ) GROUP(        ) NAME(???                 )
  LOGON/JOB INITIATION - USER AT TERMINAL         NOT RACF-DEFINED
IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND
```

- An alternate user has been requested, but the job or task user ID does not have access to the alternate user ID. For this failure, you get a violation message in the job log of the relevant queue manager.
- A context option has been used or is implied by opening a transmission queue for output, but the job user ID or, where applicable, the task or alternate user ID does not have access to the context option. In this case, a violation message is put in the job log of the relevant queue manager.
- An unauthorized user has attempted to access a secured queue manager object, for example, a queue. In this case, an ICH408I message for the violation is put in the job log of the relevant queue manager. This violation might be due to the job or, when applicable, the task or alternate user ID.

Violation messages for command security and command resource security can also be found in the job log of the queue manager.

If the ICH408I violation message shows the queue manager jobname rather than a user ID, this is normally the result of a blank alternate user ID being specified. For example:

```
ICH408I JOB(MQS1MSTR) STEP(MQS1MSTR)
          MQS1.PAYROLL.REQUEST CL(MQQUEUE)
          INSUFFICIENT ACCESS AUTHORITY
          ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

You can find out who is allowed to use blank alternate user IDs by checking the access list of the MQADMIN profile hlq.ALTERNATE.USER.-BLANK-.

An ICH408I violation message can also be generated by:

- A command being sent to the system-command input queue without context. User-written programs that write to the system-command input queue should always use a context option. For more information, see “Profiles for context security” on page 163.
- When the job accessing the MQSeries resource does not have a user ID associated with it, or when an MQSeries adapter cannot extract the user ID from the adapter environment.

| Violation messages might also be issued if you are using both queue-sharing group and queue manager level security. You might get messages indicating that no profile has been found at queue manager level, but still be granted access because of a queue-sharing group level profile.

```
ICH408I JOB(MQS1MSTR) STEP(MQS1MSTR)
         MQS1.PAYROLL.REQUEST CL(MQQUEUE)
         PROFILE NOT FOUND - REQUIRED FOR AUTHORITY CHECKING
         ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

## What to do if access is allowed or disallowed incorrectly

In addition to the steps detailed in the *Security Server (RACF) Security Administrator’s Guide*, use this checklist if access to a resource appears incorrectly controlled:

- Are the switch profiles correctly set?
  - Is RACF active?
  - Are the MQSeries RACF classes installed and active?

    Use the RACF command, SETROPTS LIST, to check this.
  - Use the MQSeries command, DISPLAY SECURITY, to display the current switch status from the queue manager.
  - Check the switch profiles in the MQADMIN class.

    Use the RACF commands, SEARCH and RLIST, for this.
  - Re-check the RACF switch profiles by issuing the MQSeries command, REFRESH SECURITY(MQADMIN).
- Has the RACF resource profile changed? For example, has universal access on the profile changed or has the access list of the profile changed?
  - Is the profile generic?

    If it is, issue the RACF command, SETROPTS GENERIC(classname) REFRESH.
  - Have you refreshed the security on this queue manager?

    | If required, issue the RACF command SETROPTS RACLIST(classname) REFRESH.

    If required, issue the MQSeries command, REFRESH SECURITY(*).
- Has the RACF definition of the user changed? For example, has the user been connected to a new group or has the user access authority been revoked?
  - Have you re-verified the user by issuing the MQSeries command, RVERIFY SECURITY(userid)?
- Are security checks being bypassed due to RESLEVEL?
  - Check the connecting user ID’s access to the RESLEVEL profile. Use the RACF audit records to determine what the RESLEVEL is set to.

- If you are running from CICS, check the transaction's RESSEC setting.
- If RESLEVEL has been changed while a user is connected, they must disconnect and re-connect before the new RESLEVEL setting takes effect.

- Are you using queue-sharing groups?
  - If you are using both queue-sharing group and queue manager level security, check that you have defined all the correct profiles. If queue manager profile is not defined, a message is sent to the log stating that the profile was not found.
  - Have you used a combination of switch settings that is not valid so that full security checking has been set on?
  - Do you need to define security switches to override some of the queue-sharing group settings for your queue manager?
  - Is a queue manager level profile taking precedence over a queue-sharing group level profile?

# Chapter 17. Security considerations for distributed queuing

This chapter discusses security considerations for distributed queuing using the channel initiator (the non-CICS mover). If you are using the CICS mover, see “Security considerations for distributed queuing (using CICS ISC)” on page 244.

This chapter also discusses security considerations for using clusters.

## The channel initiator

If you are using resource security, you should consider the following if you are using distributed queuing:

**System queues**
The channel initiator address space needs RACF UPDATE access to these system queues:
- SYSTEM.ADMIN.CHANNEL.EVENT
- SYSTEM.CHANNEL.INITQ
- SYSTEM.CHANNEL.SYNCQ
- SYSTEM.COMMAND.INPUT
- SYSTEM.COMMAND.REPLY.MODEL
- SYSTEM.QSG.CHANNEL.SYNCQ
- SYSTEM.QSG.TRANSMIT.QUEUE

and to all the user destination queues and the dead-letter queue (but see “Dead-letter queue security” on page 155).

**Transmission queues**
The channel initiator address space needs ALTER access to all the user transmission queues.

**Context security**
The channel user ID (and the MCA user ID if one has been specified) also need RACF CONTROL access to the ssid.CONTEXT profile in the MQADMIN class. Depending on the RESLEVEL profile, the network-received user ID might also need CONTROL access to this profile. See “Profiles for context security” on page 163 “RESLEVEL and channel initiator connections” on page 176 and “Chapter 15. User IDs for security checking” on page 181 for more information.

**CSQINPX**
If you are using the CSQINPX input data set, the channel initiator also needs READ access to CSQINPX, and UPDATE access to data set CSQOUTX and dynamic queues SYSTEM.CSQXCMD.*.

**Connection security**
The channel initiator address space connection requests use a connection type of CHIN, for which appropriate access security must be set, see “Connection security profiles for distributed queuing” on page 149.

**Data sets**
The channel initiator address space needs appropriate access to queue manager data sets, see “Authorizing access to data sets” on page 196.

**Commands**
The distributed queuing commands (for example, DEFINE CHANNEL,

START CHINIT, START LISTENER, and so on) should have appropriate command security set, see Table 34 on page 166.

**Channel security**
Channels, particularly receivers and server-connections, need appropriate security to be set up; see “Chapter 15. User IDs for security checking” on page 181 for more information. See also the *MQSeries Clients* manual for information about server-connection security.

**User IDs**
The user IDs described in “User IDs used by the channel initiator” on page 185 and “User IDs used by the intra-group queuing agent” on page 189 need the following:

- RACF UPDATE access to the appropriate destination queues and the dead-letter queue
- RACF CONTROL access to the `hlq.CONTEXT` profile if context checking is performed at the receiver
- Appropriate access to the hlq.ALTERNATE.USER.userid profiles they might need to use.
- For clients, the appropriate RACF access to the resources to be used.

**APPC security**
Set appropriate APPC security if you are using the LU 6.2 transmission protocol. (Use the APPCLU RACF class for example.) For information about setting up security for APPC, see the following manuals:

- *MVS Planning: APPC/MVS Management*
- *APPC Security: MVS/ESA, CICS/ESA, and OS/2* (redbook)

Outbound transmissions use the “SECURITY(SAME)” APPC option. This means that the user ID of the channel initiator address space and its default profile (RACF GROUP) are flowed across the network to the receiver with an indicator that the user ID has already been verified (ALREADYV).

If the receiving side is also OS/390, the user ID and profile are verified by APPC and the user ID is presented to the receiver channel and used as the channel user ID.

In an environment where the queue manager is using APPC to communicate with another queue manager on the same or another OS/390 system, you need to ensure that either:

- The VTAM® definition for the communicating LU specifies SETACPT(ALREADYV)
- There is a RACF APPCLU profile for the connection between LUs that specifies CONVSEC(ALREADYV)

**Changing security settings**
If the RACF access level that either the channel user ID or MCA user ID has to a destination queue is changed, this change will only take effect for new object handles (that is, new **MQOPEN**s) for the destination queue. The times when MCAs open and close queues is variable; if a channel is already running when such an access change is made, the MCA can continue to put messages on the destination queue using the existing security access of the user ID(s) rather than the updated security access. To avoid this, you should stop and re-start the channels to enforce the updated access level.

## Cluster support

This section discusses the security considerations for cluster support.

You can use the MCA user ID and security exits to authenticate cluster channels (as with conventional channels). The security exit on the cluster-receiver channel must check that the queue manager is permitted access to the server queue manager’s clusters. You can start to use MQSeries cluster support without having to change your existing queue access security, however you must allow other queue managers in the cluster to write to the SYSTEM.CLUSTER.COMMAND.QUEUE if they are to join the cluster.

MQSeries cluster support does not provide a mechanism to limit a member of a cluster to the client role only. As a result, you must be sure that you trust any queue managers that you allow into the cluster. If any queue manager in the cluster creates a queue with a particular name, it will be able to receive messages for that queue, regardless of whether the application putting messages to that queue intended this or not.

To restrict the membership of a cluster, you need to take the same action that you would take to prevent queue managers connecting to receiver channels. You can achieve this by writing a security exit program on the receiver channel or by writing an exit program to prevent unauthorized queue managers from writing to the SYSTEM.CLUSTER.COMMAND.QUEUE.

**Note:** It is not advisable to permit applications to open the SYSTEM.CLUSTER.TRANSMIT.QUEUE directly, just as it is not advisable to permit an application to open any other transmission queue directly.

If you are using resource security you should consider the following in addition to the considerations discussed in “Chapter 17. Security considerations for distributed queuing” on page 203:

**System queues**

The channel initiator needs RACF ALTER access to the following system queues:

- SYSTEM.CLUSTER.COMMAND QUEUE
- SYSTEM.CLUSTER.TRANSMIT.QUEUE.

and UPDATE access to SYSTEM.CLUSTER.REPOSITORY.QUEUE

It also needs READ access to any namelists used for clustering.

**Commands**

The cluster support commands (REFRESH and RESET CLUSTER, SUSPEND and RESUME QMGR) should have appropriate command security set (as described in Table 34 on page 166).

## Cluster security

# Chapter 18. Security considerations for using MQSeries with CICS

The CICS adapter provides the following information to MQSeries specifically for use in MQSeries security:

- Whether CICS resource-level security is active for this transaction—as specified on the RESSEC or RSLC operand of the RDO TRANSACTION definition.
- User IDs.

  For terminal tasks where a user has not signed on, the user ID is the CICS user ID associated with the terminal and is either:
  - The default CICS user ID as specified on the CICS parameter DFLTUSER SIT
  - A preset security user ID specified on the terminal definition

  For non-terminal tasks, the CICS adapter tries to get a user ID with an EXEC CICS ASSIGN command. If this is unsuccessful, the adapter tries to get the user ID using EXEC CICS INQUIRE TASK. If security is active in CICS, and the non-terminal attached transaction is defined with CMDSEC(YES), the CICS adapter passes a user ID of blanks to MQSeries.

For more information about RACF security management in the CICS environment, see the *CICS RACF Security Guide*.

## Controlling the security of CICS transactions supplied by MQSeries

The CKTI and CKAM transactions are designed to be run without a terminal; no user should have access to these transactions. These transactions are examples of what the *CICS RACF Security Guide* calls “category 1 transactions”. For information about how to set these transactions up in CICS and RACF, see the information about category 1 transactions in the *CICS RACF Security Guide*.

If you want a user to administer the CICS adapter, you must grant the user authorization these transactions:

| | |
|---|---|
| CKQC | Controls the CICS adapter functions |
| CKBM | Controls the CICS adapter functions |
| CKRT | Controls the CICS adapter functions |
| CKCN | Connect |
| CKSD | Disconnect |
| CKRS | Statistics |
| CKDP | Full screen display |
| CKDL | Line mode display |
| CKSQ | CKTI START/STOP |

If required, you can restrict access to specific functions of the adapter. For example, if you want to allow users to display the current status of the adapter via the full screen interface, but nothing else, give them access to CKQC, CKBM, CKRT, and CKDP only.

### CICS security

You should define these transactions to CICS with RESSEC(NO) and CMDSEC(NO). For more details, see the *CICS RACF Security Guide*. For information about the security of the CICS transactions supplied by MQSeries for remote queuing, see the *MQSeries Intercommunication* manual.

## CICS adapter user IDs

The user ID associated with the CICS adapter is that of the MQSeries-supplied task initiator transaction, CKTI. This section describes some of the implications of this.

### User ID checking for MQSeries resources during PLTPI and PLTSD

If an MQSeries resource is accessed during the CICS PLTPI phase, the user ID passed to MQSeries is blanks. If an MQSeries resource is accessed during the CICS PLTSD phase, the user ID passed to MQSeries is the user ID associated with the shutdown transaction.

If CKTI is started during the CICS PLTPI phase, the user ID of the CKTI task is the CICS sysidnt. This means that a user ID with the same name as the CICS sysidnt must be defined and given access to the required MQSeries resources, for example, initiation queues.

### Terminal user IDs

If CKTI is started from a terminal from the CKQC transaction or a user-written program that links to CSQCSSQ, the user ID that CKTI uses is the same as the user ID of the terminal that started CKTI.

### Automating starting of CKTI

To automate the starting of CKTIs under a specific user ID, you can use an automation product, for example, NetView. You can use this to sign on a CICS console and issue the STARTCKTI command.

You can also use preset security sequential terminals, which have been defined to emulate a CRLP terminal, with the sequential terminal input containing the CKQC STARTCKTI command.

However, when the CICS adapter alert monitor reconnects CICS to MQSeries, after, for example, an MQSeries restart, only the CKTI specified at the initial MQSeries connection is restarted. You must automate starting any extra CKTIs yourself.

### Propagating the CKTI user ID to other CICS transactions

If CKTI starts other CICS transactions, for example, message channel agents (MCAs) or user-written CICS applications, the user ID of CKTI is propagated to these applications. For example, if CKTI is running under user ID CIC1 and a trigger event occurs which requires the sender MCA transaction, CKSG, to be started, the CKSG transaction also runs under user ID CIC1. Therefore user ID CIC1 must have access to the required transmission queue.

## Security considerations for the CICS bridge

When you run the CICS bridge, you can specify the level of authentication you want to take place. If requested, the bridge checks the user ID and password extracted from the MQSeries request message before running the CICS program named in the request message.

**Notes:**

1. If you have not specified a user ID or password in a message, the bridge task runs with the LOCAL level of authentication, even if you started the bridge monitor with a different authentication option.
2. The options that include password (or passticket) validation require a CICS bridge header (MQCIH) to be provided. See the *MQSeries Application Programming Reference* manual for more information about the MQCIH header.

The level of authentication you can use is described below:

**LOCAL**
This is the default. CICS programs run by the bridge task are started with the CICS DFLTUSER user ID, therefore run with the authority associated with this user ID. There is no checking of user IDs or passwords. If a CICS program is run that tries to access protected resources, it will probably fail.

**IDENTIFY**
When you start the monitor task with the IDENTIFY authentication option, the bridge task is started with the user ID specified in the message (MQMD). CICS programs run by the bridge run with the user ID extracted from the MQMD. There is no password checking, the user ID is treated as trusted.

**VERIFY_UOW**
When you start the monitor task with the VERIFY_UOW authentication option, the monitor task checks the user ID and password by issuing the EXEC CICS VERIFY PASSWORD command before starting the bridge task. CICS programs run by the bridge run with the user ID extracted from the MQMD. If the user ID or password is invalid, the request fails with return code MQCRC_SECURITY_ERROR.

**VERIFY_ALL**
This is the same as VERIFY_UOW except that the bridge task checks the user ID and password in **every** message. This is not applicable for 3270 transactions.

If you have not specified a user ID in a message, or you have not provided a password, the CICS program started by the CICS bridge runs with the user ID set to the CICS DFLTUSER, regardless of the option requested. If you want more than one level of authentication checking performed, run a monitor task for each level you need.

Table 50 on page 210 and Table 51 on page 210 summarize the level of authority of the bridge monitor and the bridge tasks, and the use of the MQMD user ID.

## CICS security

*Table 50. CICS bridge monitor security*

| Monitor started by | At a signed on terminal | Monitor authority |
|---|---|---|
| From a terminal or EXEC CICS LINK within a program | Yes | Signed on user ID |
| From a terminal or EXEC CICS LINK within a program | No | CICS default user ID |
| EXEC CICS START with user ID | – | User ID from START |
| EXEC CICS START without user ID | – | CICS default user ID |
| The MQSeries trigger monitor CKTI | – | CICS default user ID |

*Table 51. CICS bridge task security*

| AUTH | Bridge task authority |
|---|---|
| LOCAL | CICS default user ID |
| IDENTIFY | MQMD UserIdentifier |
| VERIFY_UOW | MQMD UserIdentifier |
| VERIFY_ALL | MQMD UserIdentifier |

The options IDENTIFY, VERIFY_UOW, and VERIFY_ALL need the user ID of the bridge monitor defined to RACF as a surrogate of all the user IDs used in request messages. This is in addition to the user ID in the message being defined to RACF. (A surrogate user is one who has the authority to start work on behalf of another user, without knowing the other user’s password.)

For more information on surrogate user security, see the *CICS RACF Security Guide*.

**Note:** When IDENTIFY security is being used, you might see abend AICO for CKBP if you try to run with a user ID that has been revoked. The error reply will have return code MQCRC_BRIDGE_ERROR with reason MQFB_CICS_BRIDGE_FAILURE.

## Authority

Components of the bridge need authority to either put to or get from the various MQSeries queues. In summary:

- The monitor and all bridge tasks need authority to get messages from the bridge request queue.
- A bridge task need authority to put messages to its reply-to queue.
- To ensure any error replies are received, the monitor should have authority to put messages to all reply-to queues.
- Bridge tasks should have authority to put messages to the dead-letter queue.
- The monitor needs authority to put messages to the dead-letter queue, unless you want the bridge to stop if an error occurs.

See Table 50 to determine the correlation between user IDs and authority.

# Chapter 19. Security considerations for using MQSeries with IMS

The following section describes security considerations for using MQSeries with IMS.

## Using the OPERCMDS class

If you are using RACF to protect resources in the OPERCMDS class, ensure that your MQSeries system has authority to issue the MODIFY command to any IMS system to which it can connect.

## Security considerations for the IMS bridge

There are four aspects that you must consider when deciding your security requirements for the IMS bridge, these are:

- What security authorization is needed to connect MQSeries to IMS ("Connecting to IMS")
- How much security checking is performed on applications using the bridge to access IMS ("Application access control" on page 212)
- Which IMS resources these applications are allowed to use ("Security checking on IMS" on page 213)
- What authority is to be used for messages that are put and got by the bridge ("Security checking done by the bridge" on page 214)

When you define your security requirements for the IMS bridge you must consider the following:

- Messages passing across the bridge might have originated from applications on platforms that do not offer strong security features
- Messages passing across the bridge might have originated from applications that are not controlled by the same enterprise or organization

### Connecting to IMS

The IMS bridge is an OTMA client. The connection to IMS operates under the user ID of the MQSeries for OS/390 address space. This is normally defined as a member of the started task group. This user ID must be granted access to the OTMA group (unless the /SECURE OTMA setting is NONE).

| To do this, define the following profile in the FACILITY class:

```
IMSXCF.xcfgname.mqxcfmname
```

Where `xcfgname` is the XCF group name and `mqxcfmname` is the XCF member name of MQSeries.

You must give your MQSeries subsystem user ID read access to this profile.

**Notes:**

1. If you change the authorities in the FACILITY class, you must issue the RACF command SETROPTS RACLIST(FACILITY) REFRESH to activate the changes.

2. If profile hlq.NO.SUBSYS.SECURITY exists in the MQADMIN class, no user ID will be passed to IMS and the connection will fail unless the /SECURE OTMA setting is NONE.

## Application access control

For each IMS system that the IMS bridge connects to, you can define the following RACF profile in the FACILITY class to determine how much security checking is performed for each message passed to the IMS system.

```
IMSXCF.xcfgname.imsxcfmname
```

Where `xcfgname` is the XCF group name and `imsxcfmname` is the XCF member name for IMS. (You need to define a separate profile for each IMS system.)

The access level you allow for the MQSeries subsystem user ID in this profile is returned to MQSeries when the IMS bridge connects to IMS, and indicates the level of security that is required on subsequent transactions. For subsequent transactions, MQSeries requests the appropriate services from RACF and, where the user ID is authorized, passes the message to IMS.

OTMA does not support the IMS /SIGN command; however, MQSeries allows you to set the access checking for each message to enable implementation of the necessary level of control.

The following access level information can be returned:

**NONE or NO PROFILE FOUND**
This indicates that maximum security is required, that is, authentication is required for every transaction. A check is made to verify that the user ID specified in the *UserIdentifier* field of the MQMD structure, and the password or passticket in the *Authenticator* field of the MQIIH structure are known to RACF, and are a valid combination. A Utoken is created with a password or passticket, and passed to IMS; the Utoken is not cached.

**Note:** If profile hlq.NO.SUBSYS.SECURITY exists in the MQADMIN class, this level of security overrides whatever is defined in the profile.

**READ** This indicates that the same authentication is to be performed as above under the following circumstances:
- The first time that a specific user ID is encountered
- When the user ID has been encountered before but the cached Utoken was not created with a password or passticket

MQSeries requests a Utoken if required, and passes it to IMS.

**Note:** If a request to reverify security has been actioned, all cached information is lost and a Utoken is requested the first time each user ID is subsequently encountered.

**UPDATE**
A check is made that the user ID in the *UserIdentifier* field of the MQMD structure is known to RACF.

A Utoken is built and passed to IMS; the Utoken is cached.

**CONTROL/ALTER**
These indicate that no security Utokens need to be provided for any user IDs for this IMS system. (You would probably only use this for development and test systems.)

**Notes:**

1. This access is defined when MQSeries connects to IMS, and lasts for the duration of the connection. To change the security level, the access to the security profile must be changed and then the bridge stopped and restarted (for example, by stopping and restarting OTMA).
2. If you change the authorities in the FACILITY class, you must issue the RACF command SETROPTS RACLIST(FACILITY) REFRESH to activate the changes.
3. You can use a password or a passticket, but you must remember that the IMS bridge does not encrypt data. For information about using passtickets, see "Using RACF passtickets in the IMS header" on page 214.
4. Some of the above might be affected by security settings in IMS, using the /SECURE OTMA command.
5. Cached Utoken information is held for the duration defined by the INTERVAL and TIMEOUT parameters of the MQSeries ALTER SECURITY command.

## Security checking on IMS

Each MQSeries message that passes across the bridge contains the following security information:

- A user ID contained in the *UserIdentifier* field of the MQMD structure
- The security scope contained in the *SecurityScope* field of the MQIIH structure (if the MQIIH structure is present)
- A Utoken (unless the MQSeries sub system has CONTROL or ALTER access to the relevant `IMSXCF.xcfgname.imsxcfmname` profile)

The security checks made depend on the setting by the IMS command /SECURE OTMA, as follows:

**/SECURE OTMA NONE**
No security checks are made for the transaction.

**/SECURE OTMA CHECK**
The *UserIdentifier* field of the MQMD structure is passed to IMS for transaction or command authority checking.

An ACEE (Accessor Environment Element) is built in the IMS control region.

**/SECURE OTMA FULL**
The *UserIdentifier* field of the MQMD structure is passed to IMS for transaction or command authority checking.

An ACEE is built in the IMS dependent region as well as the IMS control region.

**/SECURE OTMA PROFILE**
The *UserIdentifier* field of the MQMD structure is passed to IMS for transaction or command authority checking

The *SecurityScope* field in the MQIIH structure is used to determine whether to build an ACEE in the IMS dependent region as well as the control region.

**Notes:**

1. If you change the authorities in the TIMS or CIMS class, or the associated group classes GIMS or DIMS, you must issue the following IMS commands to activate the changes:
   - /MODIFY PREPARE RACF
   - /MODIFY COMMIT
2. If you do not use /SECURE OTMA PROFILE, any value specified in the *SecurityScope* field of the MQIIH structure is ignored.

## Security checking done by the bridge

When the bridge puts or gets a message, the following authorities are used:

**Getting a message from the bridge queue**
No security checks are performed.

**Putting an exception, or COA report message**
Uses the authority of the user ID in the *UserIdentifier* field of the MQMD structure.

**Putting a reply message**
Uses the authority of the user ID in the *UserIdentifier* field of the MQMD structure of the original message

**Putting a message to the dead-letter queue**
No security checks are performed.

**Notes:**

1. If you change the MQSeries class profiles, you must issue the MQSeries command REFRESH SECURITY(*) to activate the changes.
2. If you change the authority of a user, you must issue the MQSeries command RVERIFY SECURITY to activate the change.

## Using RACF passtickets in the IMS header

If you want to use a passticket instead of a password in the IMS header (MQIIH), you should use an application name as if you were creating a passticket for an OS/390 batch job. That is, the APPL field should be of the form `MVSxxxx`, where xxxx is the SMFID of the OS/390 system on which the target queue manager runs.

A passticket is built from a user ID, the target application name (APPL), and a secret key. It is an 8-byte value containing uppercase alphabetic and numeric characters. It can be used only once, and is valid for a 20 minute period . For full information about passtickets, see the *Security Server (RACF) Security Administrator's Guide*.

Passtickets in IMS headers are given to RACF by MQSeries, not IMS.

# Chapter 20. Example security scenarios

This chapter describes two example security scenarios, showing the security settings required.

The first scenario uses two queue managers on OS/390, called QM1 and QM2. In the second scenario, the two queue managers are members of a queue-sharing group called QSGA. In this scenario, queue-sharing group level security is illustrated

## The two queue managers scenario

An application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The application could be a batch application or a CICS application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option. This is illustrated in Figure 30.

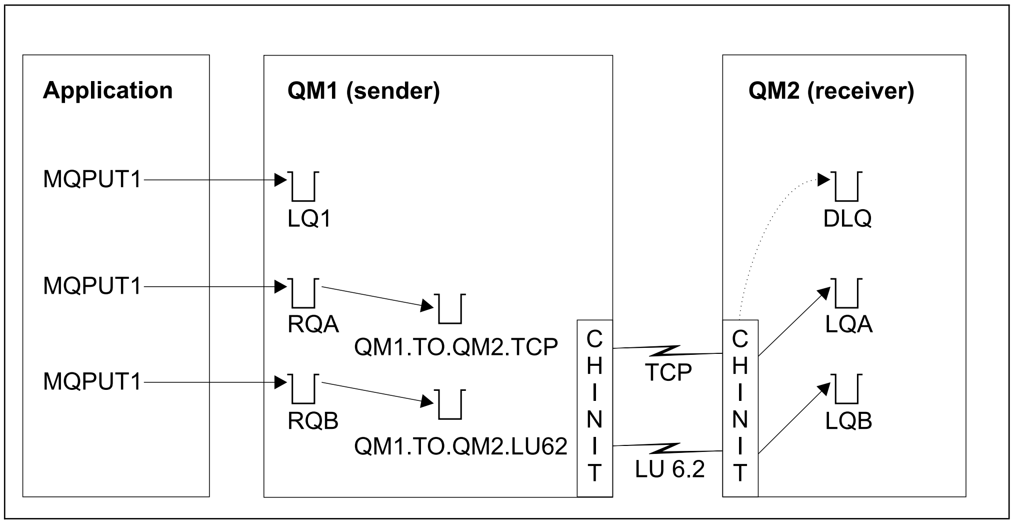

*Figure 30. Example security scenario*

The following assumptions are made about the queue managers:

- All the required MQSeries definitions have been predefined or have been made through the CSQINP2 data set processed at queue manager startup.

  If they have not, you will need the appropriate access authority to the commands needed to define these objects.
- All the RACF profiles required have been defined and appropriate access authorities have been granted, before the queue manager and channel initiators started.

  If they have not, you will need the appropriate authority to issue the RACF commands required to define all the profiles needed and grant the appropriate access authorities to those profiles. You will also need the appropriate authority to issue the MQSeries security commands to start using the new security profiles.

## Security switch settings

The following security switches are set for both queue managers:
- Subsystem security on
- Queue security on
- Alternate user security on
- Context security on
- Process security off
- Namelist security off
- Connection security on
- Command security on
- Command resource security on

The following profiles are defined in the MQADMIN class to turn process and namelist security off:

```
QM1.NO.PROCESS.CHECKS
QM1.NO.NLIST.CHECKS
QM2.NO.PROCESS.CHECKS
QM2.NO.NLIST.CHECKS
```

## MQSeries object definitions

The following objects are defined on the two queue managers. The definitions use the defaults supplied with MQSeries, unless otherwise stated.

### Queue manager QM1

The following queues are defined on queue manager QM1:

**LQ1** A local queue.

**RQA** A remote queue definition, with the following attributes:
- RNAME(LQA)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TCP)

**RQB** A remote queue definition, with the following attributes:
- RNAME(LQB)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.LU62)

**QM1.TO.QM2.TCP**
A transmission queue.

**QM1.TO.QM2.LU62**
A transmission queue.

The following channels are defined on QM1:

**QM1.TO.QM2.TCP**
A sender channel definition, with the following attributes:
- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TCP)
- CONNAME(QM2TCP)

**QM1.TO.QM2.LU62**

A sender channel definition, with the following attributes:
- CHLTYPE(SDR)
- TRPTYPE(LU62)
- XMITQ(QM1.TO.QM2.LU62)
- CONNAME(QM2LU62)

(See "Chapter 17. Security considerations for distributed queuing" on page 203 for information about setting up APPC security.)

### Queue manager QM2

The following queues have been defined on queue manager QM2:

| | |
|---|---|
| **LQA** | A local queue. |
| **LQB** | A local queue. |
| **DLQ** | A local queue that is used as the dead-letter queue. |

The following channels have been defined on QM2:

**QM1.TO.QM2.TCP**

A receiver channel definition, with the following attributes:
- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCATCP)

**QM1.TO.QM2.LU62**

A receiver channel definition, with the following attributes:
- CHLTYPE(RCVR)
- TRPTYPE(LU62)
- PUTAUT(CTX)
- MCAUSER(MCALU62)

(See "Chapter 17. Security considerations for distributed queuing" on page 203 for information about setting up APPC security.)

## User IDs used in scenario

The following user IDs are used:

| | |
|---|---|
| **BATCHID** | Batch application (Job or TSO ID) |
| **MSGUSR** | *UserIdentifier* in MQMD (context user ID) |
| **MOVER1** | QM1 channel initiator address space user ID |
| **MOVER2** | QM2 channel initiator address space user ID |
| **MCATCP** | MCAUSER specified on the TCP/IP receiver channel definition |
| **MCALU62** | MCAUSER specified on the LU 6.2 receiver channel definition |
| **CICSAD1** | CICS address space ID |
| **CICSTX1** | CICS task user ID |

## Security profiles and accesses required

Table 52 through Table 56 on page 220 show the security profiles that are required to enable the scenario to work:

Table 52. Security profiles for the example scenario

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQCONN | QM1.CHIN | MOVER1 | READ |
| MQADMIN | QM1.RESLEVEL | BATCHID | NONE |
| | | CICSAD1 | NONE |
| | | MOVER1 | NONE |
| MQADMIN | QM1.CONTEXT | MOVER1 | CONTROL |
| MQQUEUE | QM1.SYSTEM.COMMAND.INPUT | MOVER1 | UPDATE |
| MQQUEUE | QM1.SYSTEM.CHANNEL.SYNCQ | MOVER1 | UPDATE |
| MQQUEUE | QM1.SYSTEM.CHANNEL.INITQ | MOVER1 | UPDATE |
| MQQUEUE | QM1.SYSTEM.COMMAND.REPLY.MODEL | MOVER1 | UPDATE |
| MQQUEUE | QM1.SYSTEM.ADMIN.CHANNEL.EVENT | MOVER1 | UPDATE |
| MQQUEUE | QM1.QM1.TO.QM2.TCP | MOVER1 | ALTER |
| MQQUEUE | QM1.QM1.TO.QM2.LU62 | MOVER1 | ALTER |
| MQCONN | QM2.CHIN | MOVER2 | READ |
| MQADMIN | QM2.RESLEVEL | MOVER2 | NONE |
| MQADMIN | QM2.CONTEXT | MOVER2 | CONTROL |
| MQQUEUE | QM2.SYSTEM.COMMAND.INPUT | MOVER2 | UPDATE |
| MQQUEUE | QM2.SYSTEM.CHANNEL.SYNCQ | MOVER2 | UPDATE |
| MQQUEUE | QM2.SYSTEM.CHANNEL.INITQ | MOVER2 | UPDATE |
| MQQUEUE | QM2.SYSTEM.COMMAND.REPLY.MODEL | MOVER2 | UPDATE |
| MQQUEUE | QM2.SYSTEM.ADMIN.CHANNEL.EVENT | MOVER2 | UPDATE |
| MQQUEUE | QM2.DLQ | MOVER2 | UPDATE |

### Security profiles required for a batch application

The batch application runs under user ID BATCHID on QM1. It connects to queue manager QM1 and puts messages to the following queues:

- LQ1
- RQA
- RQB

It uses the MQPMO_SET_ALL_CONTEXT and MQPMO_ALTERNATE_USER_AUTHORITY options. The alternate user ID found in the *UserIdentifier* field of the message descriptor (MQMD) is MSGUSR.

The following profiles are required on queue manager QM1:

Table 53. Sample security profiles for the batch application on queue manager QM1

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQCONN | QM1.BATCH | BATCHID | READ |
| MQADMIN | QM1.CONTEXT | BATCHID | CONTROL |
| MQQUEUE | QM1.LQ1 | BATCHID | UPDATE |
| MQQUEUE | QM1.RQA | BATCHID | UPDATE |
| MQQUEUE | QM1.RQB | BATCHID | UPDATE |

The following profiles are required on queue manager QM2 for messages put to queue RQA on queue manager QM1 (for the TCP/IP channel):

Table 54. Sample security profiles for queue manager QM2 using TCP/IP

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQADMIN | QM2.ALTERNATE.USER.MSGUSR | MCATCP | UPDATE |
| | | MOVER2 | UPDATE |
| MQADMIN | QM2.CONTEXT | MCATCP | CONTROL |
| | | MOVER2 | CONTROL |
| MQQUEUE | QM2.LQA | MOVER2 | UPDATE |
| | | MSGUSR | UPDATE |
| MQQUEUE | QM2.DLQ | MOVER2 | UPDATE |
| | | MSGUSR | UPDATE |

**Notes:**

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCATCP).
2. The MCAUSER field of the receiver channel definition is set to MCATCP; this user ID is used in addition to the channel initiator address space user ID for the checks carried out against the alternate user ID and context profile.
3. The MOVER2 user ID and the *UserIdentifier* in the message descriptor (MQMD) are used for the resource checks against the queue.
4. The MOVER2 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.
5. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.

## Example security scenarios

The following profiles are required on queue manager QM2 for messages put to queue RQB on queue manager QM1 (for the LU 6.2 channel):

*Table 55. Sample security profiles for queue manager QM2 using LU 6.2*

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQADMIN | QM2.ALTERNATE.USER.MSGUSR | MCALU62 | UPDATE |
| | | MOVER1 | UPDATE |
| MQADMIN | QM2.CONTEXT | MCALU62 | CONTROL |
| | | MOVER1 | CONTROL |
| MQQUEUE | QM2.LQB | MOVER1 | UPDATE |
| | | MSGUSR | UPDATE |
| MQQUEUE | QM2.DLQ | MOVER1 | UPDATE |
| | | MSGUSR | UPDATE |

**Notes:**

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCALU62).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCALU62).
3. Because LU 6.2 supports security on the communications system for the channel, the user ID received from the network is used as the channel user ID (MOVER1).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCALU62 and MOVER1 are used for the checks performed against the alternate user ID and Context profiles, and MCALU62 and MOVER1 are used for the checks against the queue profile.
6. The MOVER1 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

## Security profiles required for a CICS application

The CICS application uses a CICS address space user ID of CICSAD1 and a CICS task user ID of CICSTX1. The security profiles required on queue manager QM1 are different to those required for the batch application. The profiles required on queue manager QM2 are the same as for the batch application.

The following profiles are required on queue manager QM1:

*Table 56. Sample security profiles for the CICS application on queue manager QM1*

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQCONN | QM1.CICS | CICSAD1 | READ |
| MQADMIN | QM1.CONTEXT | CICSAD1 | CONTROL |
| | | CICSTX1 | CONTROL |
| MQQUEUE | QM1.LQ1 | CICSAD1 | UPDATE |
| | | CICSTX1 | UPDATE |
| MQQUEUE | QM1.RQA | CICSAD1 | UPDATE |
| | | CICSTX1 | UPDATE |

*Table 56. Sample security profiles for the CICS application on queue manager QM1 (continued)*

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQQUEUE | QM1.RQB | CICSAD1 | UPDATE |
| | | CICSTX1 | UPDATE |

## The queue-sharing group scenario

An application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The application is a batch application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option. This is illustrated in Figure 30 on page 215.

The following assumptions are made about the queue managers:

- All the required MQSeries definitions have been predefined or have been made through the CSQINP2 data set processed at queue manager startup.

  If they have not, you will need the appropriate access authority to the commands needed to define these objects.
- All the RACF profiles required have been defined and appropriate access authorities have been granted, before the queue manager and channel initiators started.

  If they have not, you will need the appropriate authority to issue the RACF commands required to define all the profiles needed and grant the appropriate access authorities to those profiles. You will also need the appropriate authority to issue the MQSeries security commands to start using the new security profiles.

### Security switch settings

The following security switches are set for the queue-sharing group:
- Subsystem security on
- Queue-sharing group security on
- Queue manager security off
- Queue security on
- Alternate user security on
- Context security on
- Process security off
- Namelist security off
- Connection security on
- Command security on
- Command resource security on

The following profiles are defined in the MQADMIN class to turn process, namelist, and queue-manager level security off:

```
QSGA.NO.PROCESS.CHECKS
QSGA.NO.NLIST.CHECKS
QSGA.NO.QMGR.CHECKS
```

## MQSeries object definitions

The following objects are defined on the two queue managers. The definitions use the defaults supplied with MQSeries, unless otherwise stated.

### Queue manager QM1

The following queues are defined on queue manager QM1:

**LQ1**
A local queue.

**RQA**
A remote queue definition, with the following attributes:
- RNAME(LQA)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TCP)

**RQB**
A remote queue definition, with the following attributes:
- RNAME(LQB)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.LU62)

**QM1.TO.QM2.TCP**
A transmission queue.

**QM1.TO.QM2.LU62**
A transmission queue.

The following channels are defined on QM1:

**QM1.TO.QM2.TCP**
A sender channel definition, with the following attributes:
- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TCP)
- CONNAME(QM2TCP)

**QM1.TO.QM2.LU62**
A sender channel definition, with the following attributes:
- CHLTYPE(SDR)
- TRPTYPE(LU62)
- XMITQ(QM1.TO.QM2.LU62)
- CONNAME(QM2LU62)

(See “Chapter 17. Security considerations for distributed queuing” on page 203 for information about setting up APPC security.)

### Queue manager QM2

The following queues have been defined on queue manager QM2:

**LQA** A local queue.
**LQB** A local queue.
**DLQ** A local queue that is used as the dead-letter queue.

The following channels have been defined on QM2:

**QM1.TO.QM2.TCP**
A receiver channel definition, with the following attributes:
- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCATCP)

**QM1.TO.QM2.LU62**
A receiver channel definition, with the following attributes:
- CHLTYPE(RCVR)
- TRPTYPE(LU62)
- PUTAUT(CTX)
- MCAUSER(MCALU62)

(See "Chapter 17. Security considerations for distributed queuing" on page 203 for information about setting up APPC security.)

## User IDs used in scenario

The following user IDs are used:

**BATCHID**
Batch application (Job or TSO ID)

**MSGUSR**
*UserIdentifier* in MQMD (context user ID)

**MOVER1**
QM1 channel initiator address space user ID

**MOVER2**
QM2 channel initiator address space user ID

**MCATCP**
MCAUSER specified on the TCP/IP receiver channel definition

**MCALU62**
MCAUSER specified on the LU 6.2 receiver channel definition

## Security profiles and accesses required

Table 57 through Table 60 on page 226 show the security profiles that are required to enable the scenario to work:

*Table 57. Security profiles for the example scenario*

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQCONN | QSGA.CHIN | MOVER1 | READ |
| | | MOVER2 | READ |
| MQADMIN | QSGA.RESLEVEL | BATCHID | NONE |
| | | MOVER1 | NONE |
| | | MOVER2 | NONE |
| MQADMIN | QSGA.CONTEXT | MOVER1 | CONTROL |
| | | MOVER2 | CONTROL |
| MQQUEUE | QSGA.SYSTEM.COMMAND.INPUT | MOVER1 | UPDATE |
| | | MOVER2 | UPDATE |
| MQQUEUE | QSGA.SYSTEM.CHANNEL.SYNCQ | MOVER1 | UPDATE |
| | | MOVER | UPDATE |
| MQQUEUE | QSGA.SYSTEM.CHANNEL.INITQ | MOVER1 | UPDATE |
| | | MOVER2 | UPDATE |
| MQQUEUE | QSGA.SYSTEM.COMMAND.REPLY.MODEL | MOVER1 | UPDATE |
| | | MOVER2 | UPDATE |
| MQQUEUE | QSGA.SYSTEM.ADMIN.CHANNEL.EVENT | MOVER1 | UPDATE |
| | | MOVER2 | UPDATE |

Table 57. Security profiles for the example scenario (continued)

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQQUEUE | QSGA.SYSTEM.QSG.CHANNEL.SYNCQ | MOVER1 | UPDATE |
| | | MOVER2 | UPDATE |
| MQQUEUE | QSGA.SYSTEM.QSG.TRANSMIT.QUEUE | MOVER1 | UPDATE |
| | | MOVER2 | UPDATE |
| MQQUEUE | QSGA.QM1.TO.QM2.TCP | MOVER1 | ALTER |
| MQQUEUE | QSGA.QM1.TO.QM2.LU62 | MOVER1 | ALTER |
| MQQUEUE | QSGA.DLQ | MOVER2 | UPDATE |

## Security profiles required for a batch application

The batch application runs under user ID BATCHID on QM1. It connects to queue manager QM1 and puts messages to the following queues:
- LQ1
- RQA
- RQB

It uses the MQPMO_SET_ALL_CONTEXT and MQPMO_ALTERNATE_USER_AUTHORITY options. The alternate user ID found in the *UserIdentifier* field of the message descriptor (MQMD) is MSGUSR.

The following profiles are required on queue manager QM1:

Table 58. Sample security profiles for the batch application on queue manager QM1

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQCONN | QSGA.BATCH | BATCHID | READ |
| MQADMIN | QSGA.CONTEXT | BATCHID | CONTROL |
| MQQUEUE | QSGA.LQ1 | BATCHID | UPDATE |
| MQQUEUE | QSGA.RQA | BATCHID | UPDATE |
| MQQUEUE | QSGA.RQB | BATCHID | UPDATE |

The following profiles are required on queue manager QM2 for messages put to queue RQA on queue manager QM1 (for the TCP/IP channel):

Table 59. Sample security profiles for queue manager QM2 using TCP/IP

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQADMIN | QSGA.ALTERNATE.USER.MSGUSR | MCATCP | UPDATE |
| | | MOVER2 | UPDATE |
| MQADMIN | QSGA.CONTEXT | MCATCP | CONTROL |
| | | MOVER2 | CONTROL |
| MQQUEUE | QSGA.LQA | MOVER2 | UPDATE |
| | | MSGUSR | UPDATE |
| MQQUEUE | QSGA.DLQ | MOVER2 | UPDATE |
| | | MSGUSR | UPDATE |

**Notes:**

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCATCP).
2. The MCAUSER field of the receiver channel definition is set to MCATCP; this user ID is used in addition to the channel initiator address space user ID for the checks carried out against the alternate user ID and context profile.
3. The MOVER2 user ID and the *UserIdentifier* in the message descriptor (MQMD) are used for the resource checks against the queue.
4. The MOVER2 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.
5. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.

The following profiles are required on queue manager QM2 for messages put to queue RQB on queue manager QM1 (for the LU 6.2 channel):

*Table 60. Sample security profiles for queue manager QM2 using LU 6.2*

| Class | Profile | User ID | Access |
|---|---|---|---|
| MQADMIN | QSGA.ALTERNATE.USER.MSGUSR | MCALU62 | UPDATE |
| | | MOVER1 | UPDATE |
| MQADMIN | QSGA.CONTEXT | MCALU62 | CONTROL |
| | | MOVER1 | CONTROL |
| MQQUEUE | QSGA.LQB | MOVER1 | UPDATE |
| | | MSGUSR | UPDATE |
| MQQUEUE | QSGA.DLQ | MOVER1 | UPDATE |
| | | MSGUSR | UPDATE |

**Notes:**

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCALU62).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCALU62).
3. Because LU 6.2 supports security on the communications system for the channel, the user ID received from the network is used as the channel user ID (MOVER1).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCALU62 and MOVER1 are used for the checks performed against the alternate user ID and Context profiles, and MCALU62 and MOVER1 are used for the checks against the queue profile.
6. The MOVER1 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

# Chapter 21. MQSeries security implementation checklist

This chapter gives a step-by-step procedure you can use to work out and define the security implementation for each of your MQSeries subsystems. Refer to other sections for details, in particular “Chapter 13. Profiles used to control access to MQSeries resources” on page 147.

If you require security checking, follow this checklist to implement it:

1. Activate the RACF MQADMIN class.
2. Do you want security at queue-sharing group level, queue-manager level, or a combination of both?

   Refer to “Profiles to control queue-sharing group or queue manager level security” on page 141.
3. Do you need connection security?

   **Yes**: Activate the MQCONN class. Define appropriate connection profiles at either queue manager level or queue-sharing group level in the MQCONN class and permit the appropriate users or groups access to these profiles.

   **Note:** Only users of the **MQCONN** API request or CICS or IMS address space user IDs need to have access to the corresponding connection profile.

   **No**: Define an hlq.NO.CONNECT.CHECKS profile at either queue manager level or queue-sharing group level in the MQADMIN class.
4. Do you need security checking on commands?

   **Yes**: Activate the MQCMDS class. Define appropriate command profiles at either queue manager level or queue-sharing group level in the MQCMDS class and permit the appropriate users or groups access to these profiles.

   **No**: Define an hlq.NO.CMD.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN class.
5. Do you need security on the resources used in commands?

   **Yes**: Ensure the MQADMIN class is active. Define appropriate profiles for protecting resources on commands at either queue manager level or queue-sharing group level in the MQADMIN class and permit the appropriate users or groups access to these profiles. Set the CMDUSER parameter in CSQ6SYSP to the default user ID to be used for command security checks.

   **No**: Define an hlq.NO.CMD.RESC.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN class.
6. Do you need queue security?

   **Yes**: Activate the MQQUEUE class. Define appropriate queue profiles for the required queue manager or queue-sharing group in the MQQUEUE class and permit the appropriate users or groups access to these profiles.

   **No**: Define an hlq.NO.QUEUE.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN class.

7. Do you need process security?

   **Yes**: Activate the MQPROC class. Define appropriate process profiles at either queue manager or queue-sharing group level and permit the appropriate users or groups access to these profiles.

   **No**: Define an hlq.NO.PROCESS.CHECKS profile for the appropriate queue manager or queue-sharing group in the MQADMIN class.

8. Do you need namelist security?

   **Yes**: Activate the MQNLIST class. Define appropriate namelist profiles at either queue manager level or queue-sharing group level in the MQNLIST class and permit the appropriate users or groups access to these profiles.

   **No**: Define an hlq.NO.NLIST.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN class.

9. Do any users need to protect the use of the **MQOPEN** or **MQPUT1** options relating to the use of context?

   **Yes**: Ensure the MQADMIN class is active. Define an hlq.CONTEXT profile at either queue manager level or queue-sharing group level in the MQADMIN class and permit the appropriate users or groups access to this profile.

   **No**: Define an hlq.NO.CONTEXT.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN class.

10. Do you need to protect the use of alternate user IDs?

    **Yes**: Ensure the MQADMIN class is active. Define the appropriate hlq.ALTERNATE.USER.*alternateuserid* profiles for the required queue manager or queue-sharing group and permit the required users or groups access to these profiles.

    **No**: Define the profile hlq.NO.ALTERNATE.USER.CHECKS for the required queue manager or queue-sharing group in the MQADMIN class.

11. Do you need to tailor which user IDs are to be used for resource security checks through RESLEVEL?

    **Yes**: Ensure the MQADMIN class is active. Define an hlq.RESLEVEL profile at either queue manager level or queue-sharing group level in the MQADMIN class and permit the required users or groups access to the profile.

    **No**: Ensure that no generic profiles exist in the MQADMIN class that could apply to hlq.RESLEVEL. Define an hlq.RESLEVEL profile for the required queue manager or queue-sharing group and ensure that no users or groups have access to it.

12. Do you need to 'time out' unused user IDs from MQSeries?

    **Yes**: Determine what timeout values you would like to use and issue the MQSeries ALTER SECURITY command to change the TIMEOUT and INTERVAL parameters.

    **No**: Issue the MQSeries ALTER SECURITY command to set the INTERVAL value to zero.

**Note:** Update the CSQINP1 data set used by your subsystem so that the MQSeries ALTER SECURITY command is issued automatically at every MQSeries start up.

13. Do you use distributed queuing (without CICS)?

    **Yes**: Determine the appropriate MCAUSER attribute value for each channel and provide suitable channel security exits if required.

14. Do you use clients?

    **Yes**: Determine the appropriate MCAUSER attribute value for each server-connection channel, and provide suitable channel security exits if required.

| 15. Check your switch settings.

| MQSeries issues messages at queue manager startup that display your
| security settings. Use these messages to determine whether your switches are
| set correctly. For an example of these messages, see the *MQSeries for OS/390*
| *System Administration Guide*.

# Part 6. Appendixes

# Appendix A. Upgrading and applying service to TCP/IP, Language Environment, or OS/390 Callable Services

The following tables show you what you need to do to MQSeries for OS/390 if you upgrade your level of, or apply service to, the following products:

- TCP/IP
- Language Environment®
- OS/390 Callable Services (APPC and RRS for example)

*Table 61. Service has been applied or the product has been upgraded to a new release*

| Product | Action if using CALLLIBS | Action if using LINK |
|---|---|---|
| TCP/IP | You need to do this only if the TCP/IP module DSPREFIX in the SEZACMTX library has been changed.<br>1. Run REPORT CALLLIBS for DDDEF SEZACMTX.<br>2. Run the job generated by REPORT CALLLIBS. | No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8LDQM job has been run. |
| Language Environment | 1. Run REPORT CALLLIBS for DDDEFs SCEELKED and SCEESPC.<br>2. Run the job generated by REPORT CALLLIBS. | No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8LDQM job has been run. |
| Callable Services | 1. Run REPORT CALLLIBS for DDDEF CSSLIB.<br>2. Run the job generated by REPORT CALLLIBS. | No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8LDQM job has been run. |

*Table 62. One of the products has been updated to a new release in a new SMP/E environment and libraries*

| Product | Action if using CALLLIBS | Action if using LINK |
|---|---|---|
| TCP/IP | 1. Change the DDDEF for SEZACMTX to point to the new library.<br>2. Run REPORT CALLLIBS for DDDEF SEZACMTX.<br>3. Run the job generated by REPORT CALLLIBS. | 1. Delete the XZMOD subentries for the following LMOD entries in the MQSeries for OS/390 target zone:<br>• CSQXRCTL<br>• CSQXSUPR<br>• CSQXTCMI<br>• CSQXTCP<br>2. Set up the appropriate ZONEINDEXs between the MQSeries zones and the TCP/IP zone.<br>3. Tailor CSQ8LDQM to refer to the new zone on the FROMZONE parameter of the LINK commands (CSQ8LDQM can be found in the SCSQINST library).<br>4. Run CSQ8LDQM. |

*Table 62. One of the products has been updated to a new release in a new SMP/E environment and libraries (continued)*

| Product | Action if using CALLLIBS | Action if using LINK |
|---|---|---|
| Language Environment | 1. Change the DDDEFs for SCEELKED and SCEESPC to point to the new library.<br>2. Run REPORT CALLLIBS for DDDEFs SCEELKED and SCEESPC.<br>3. Run the job generated by REPORT CALLLIBS. | 1. Delete the XZMOD subentries for the following LMOD entries in the MQSeries for OS/390 target zone:<br>AMTASM10, AMTBL10, AMTBS10, AMTCL10, AMTCS10, AMTIL10, AMTIS10, AMTRL10, AMTRS10, CMQXDCST, CMQXRCTL, CMQXSUPR, CSQCBE00, CSQCBE30, CSQCBP00, CSQCBP10, CSQCBR00, CSQUCVX, CSQUDLQH, CSQVXPCB, CSQVXSPT, CSQXDCST, CSQXRCTL, CSQXSUPR, CSQXTCMI, CSQXTCP, CSQXTNSV, CSQ7DRPS, IMQB23IC, IMQB23IM, IMQB23IR, IMQS23IC, IMQS23IM, IMQS23IR<br>2. Set up the appropriate ZONEINDEXs between the MQSeries zones and the Language Environment zones.<br>3. Tailor CSQ8LDQM to refer to the new zone on the FROMZONE parameter of the LINK commands. CSQ8LDQM can be found in the SCSQINST library.<br>4. Run CSQ8LDQM. |
| Callable services | 1. Change the DDDEF for CSSLIB to point to the new library.<br>2. Run REPORT CALLLIBS for DDDEF CSSLIB.<br>3. Run the job generated by REPORT CALLLIBS. | 1. Delete the XZMOD subentries for the following LMOD entries in the MQSeries for OS/390 target zone:<br>CMQXRCTL, CMQXSUPR, CSQBSRV, CSQILPLM, CSQXJST, CSQXRCTL, CSQXSUPR, CSQ3AMGP, CSQ3EPX, CSQ3REPL<br>2. Set up the appropriate ZONEINDEXs between the MQSeries zones and the Callable Services zones.<br>3. Tailor CSQ8LDQM to refer to the new zone on the FROMZONE parameter of the LINK commands. CSQ8LDQM can be found in the SCSQINST library.<br>4. Run CSQ8LDQM. |

## Running a REPORT CALLLIBS job

When running a REPORT CALLLIBS job there is an option that instructs SMP/E to place a user created job card at the front of the output data set. The option (JOBCARD) requires a DDDEF that points to a PDS containing the job card. This DDDEF can be defined to SMP/E using the example job step shown in Figure 31.

```
//******************************************************************
//* DEFINE THE LOCATION TO SMP/E OF THE JOBCARD CALLLIBS IS TO
//* USE WHEN CREATING THE LINK-EDIT JOB.
//******************************************************************
//DEFJCARD  EXEC PGM=GIMSMP,REGION=4096K
//SMPCSI   DD DSN=your.csi,
//            DISP=SHR
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
  SET BDY(GLOBAL).
  UCLIN .
    REP DDDEF(MQJCARD)
        DA(job.card.data.set) SHR .
  ENDUCL .
/*
```

*Figure 31. Example SMP/E JOBCARD job step*

To run a REPORT CALLLIBS job you need to supply the following:
- The data set names of SMP/E CSI that contains MQSeries for OS/390
- The data set into which the linkedit job is to be placed

In the SMP/E commands you will need to provide:
- The DDDEFs of the products that have been upgraded or updated (see Table 61 on page 233 and Table 62 on page 233 for a list of possible DDDEFs)
- The DDDEF of the PDS containing the JOBCARD (MQJCARD in the example below)
- The name of the member containing the JOBCARD

Figure 32 shows an example SMP/E job that can used as a basis for your own job.

```
//******************************************************************
//* RUN REPORT CALLLIBS.
//******************************************************************
//CALLLIBS  EXEC PGM=GIMSMP,REGION=4096K
//SMPCSI   DD DSN=your.csi,
//            DISP=SHR
//SMPPUNCH DD DSN=your.callibs.punched.output,
//            DISP=(MOD,CATLG),
//            UNIT=SYSDA,
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=8800),
//            SPACE=(8800,(10,2))
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
  SET BDY(GLOBAL).
  REPORT CALLLIBS(dddef)
  ZONES(target zone)
  JOBCARD(MQJCARD,jobcard member name).
/*
```

*Figure 32. Example SMP/E REPORT CALLLIBS job*

# Appendix B. Using OTMA exits in IMS

If you want to send output from an IMS transaction to MQSeries, and that transaction did not originate in MQSeries, you need to code one or more IMS OTMA exits.

Similarly if you want to send output to a non-OTMA destination, and the transaction did originate in MQSeries, you also need to code one or more IMS OTMA exits.

The following exits are available in IMS to enable you to customize processing between IMS and MQSeries:
- An OTMA pre-routing exit
- A destination resolution user (DRU) exit

## Exit names

You must name the pre-routing exit DFSYPRX0. You can name the DRU exit anything, as long as it does not conflict with a module name already in IMS.

### Specifying the destination resolution user exit name

You can use the *Druexit* parameter of the OTMACON keyword of the CSQ6SYSP macro to specify the name of the OTMA DRU exit to be run by IMS.

We suggest you adopt a naming convention of DRU0xxxx, where xxxx is the name of your MQSeries system.

If you do not specify the name of a DRU exit in the OTMACON parameter, the default is DFSYDRU0. A sample of this module is supplied by IMS. See the *IMS/ESA Customization Guide* for information about this.

### Naming convention for IMS destination

You need a naming convention for the destination to which you send the output from your IMS program. This is the destination that is set in the CHNG call of your IMS application, or that is pre-set in the IMS PSB.

## A sample scenario

We suggest the OTMA destination name is synonymous with the MQSeries system name, for example the MQSeries system name repeated. (In this case, if the MQSeries system name is VCPE, the destination set by the CHNG call is VCPEVCPE.)

### The pre-routing exit DFSYPRX0

You must first code a pre-routing exit DFSYPRX0. Parameters passed to this routine by IMS are documented in *IMS/ESA Customization Guide.*

This exit tests whether the message is intended for a known OTMA destination (in our example VCPEVCPE). If it is, the exit must check whether the transaction sending the message originated in OTMA. If so, it will already have an OTMA header, so you should exit from DFSYPRX0 with register 15 set to 0.

## IMS OTMA exits

- If the transaction sending the message did not originate in OTMA, you must set the client name to be a valid OTMA client. This is the XCF member-name of the MQSeries system to which you want to send the message. The *IMS/ESA Customization Guide* tells you where to set this. We suggest you set your client name (in the OTMACON parameter of the CSQ6SYSP macro) to be the queue manager name. This is the default. You should then exit from DFSYPRX0 setting register 15 to 4.
- If the transaction sending the message originated in OTMA, and the destination is non-OTMA, you should set register 15 to 8 and exit.
- In all other cases, you should set register 15 to 0.

If you set the OTMA client name to one that is not known to IMS, your application CHNG or ISRT call returns an A1 status code.

For an IMS system communicating with more than one MQSeries system, you should repeat the logic above for each MQSeries system.

Sample assembler code to achieve the above is shown in Figure 33:

```
         TITLE 'DFSYPRX0: OTMA PRE-ROUTING USER EXIT'
DFSYPRX0 CSECT
DFSYPRX0 AMODE 31
DFSYPRX0 RMODE ANY
*
         SAVE  (14,12),,DFSYPRX0&SYSDATE&SYSTIME
         SPACE 2
         LR    R12,R15                MODULE ADDRESSABILITY
         USING DFSYPRX0,R12
*
         L     R2,12(,R1)             R2 -> OTMA PREROUTE PARMS
*
         LA    R3,48(,R2)             R3 AT ORIGINAL OTMA CLIENT (IF ANY)
         CLC   0(16,R3),=XL16'00'     OTMA ORIG?
         BNE   OTMAIN                 YES, GO TO THAT CODE
*
NOOTMAIN DS 0H                        NOT OTMA INPUT
         LA    R5,8(,R2)              R5 IS AT THE DESTINATION NAME
         CLC   0(8,R5),=C'VCPEVCPE'   IS IT THE OTMA UNSOLICITED DEST?
         BNE   EXIT0                  NO, NORMAL PROCESSING
*
         L     R4,80(,R2)             R4 AT ADDR OF OTMA CLIENT
         MVC   0(16,R4),=CL16'VCPE'   CLIENT OVERRIDE
         B     EXIT4                  AND EXIT
*
OTMAIN   DS 0H                        OTMA INPUT
         LA    R5,8(,R2)              R5 IS AT THE DESTINATION NAME
         CLC   0(8,R5),=C'VCPEVCPE'   IS IT THE OTMA UNSOLICITED DEST?
         BNE   EXIT8                  NO, NORMAL PROCESSING
```

*Figure 33. OTMA pre-routing exit assembler sample (Part 1 of 2)*

```
*
EXIT0    DS 0H
         LA    R15,0                RC = 0
         B     BYEBYE
*
EXIT4    DS 0H
         LA    R15,4                RC = 4
         B     BYEBYE
*
EXIT8    DS 0H
         LA    R15,8                RC = 8
         B     BYEBYE
*
BYEBYE   DS 0H
         RETURN (14,12),,RC=(15)    RETURN WITH RETURN CODE IN R15
         SPACE 2
         REQUATE
         SPACE 2
         END
```

*Figure 33. OTMA pre-routing exit assembler sample (Part 2 of 2)*

## The destination resolution user exit

If you have set register 15 to 4 in DFSYPRX0, or if the source of the transaction was OTMA ***and*** you set Register 15 to 0, your DRU exit is invoked. In our example, the DRU exit name is DRU0VCPE.

The DRU exit checks if the destination is VCPEVCPE. If it is, it sets the OTMA user data (in the OTMA prefix) as follows:

| Offset (decimal) | OTMA user data |
|---|---|
| **0** | OTMA user data length (in this example, 334) |
| **2** | MQMD |
| **326** | Reply to format |

These offsets are where the MQSeries-IMS bridge expects to find this information.

We suggest that the DRU exit is as simple as possible. Therefore, in this sample, all messages originating in IMS for a particular MQSeries system will be put to the same MQSeries queue.

If the message needs to be persistent, IMS must use a synchronized transaction pipe. To do this, the DRU exit must set the OUTPUT flag. For further details, please refer to the *IMS/ESA Customization Guide.*

You should write an MQSeries application to process this queue, and use information from the MQMD structure, the MQIIH structure (if present), or the user data, to route each message to its destination.

A sample assembler DRU exit is shown in Figure 34 on page 240.

```
         TITLE 'DRU0VCPE: OTMA DESTINATION RESOLUTION USER EXIT'
DRU0VCPE CSECT
DRU0VCPE AMODE 31
DRU0VCPE RMODE ANY
*
         SAVE  (14,12),,DRU0VCPE&SYSDATE&SYSTIME
         SPACE 2
         LR    R12,R15                       MODULE ADDRESSABILITY
         USING DRU0VCPE,R12
*
         L     R2,12(,R1)                    R2 -> OTMA DRU PARMS
*
         L     R5,88(,R2)                    R5 ADDR OF OTMA USERDATA
         LA    R6,2(,R5)                     R6 ADDR OF MQMD
         USING MQMD,R6                       AS A BASE
*
         LA    R4,MQMD_LENGTH+10             SET THE OTMA USERDATA LEN
         STH   R4,0(,R5)                     = LL + MQMD + 8
*                                            CLEAR REST OF USERDATA
         MVI   0(R6),X'00'                   ...NULL FIRST BYTE
         MVC   1(255,R6),0(R6)               ...AND PROPAGATE IT
         MVC   256(MQMD_LENGTH-256+8,R6),255(R6) ...AND PROPAGATE IT
*
VCPE     DS    0H
         CLC   44(16,R2),=CL16'VCPE'         IS DESTINATION VCPE?
         BNE   EXIT4                         NO, THEN DEST IS NON-OTMA
         MVC   MQMD_REPLYTOQ,=CL48'IMS.BRIDGE.UNSOLICITED.QUEUE'
         MVC   MQMD_REPLYTOQMGR,=CL48'VCPE'   SET QNAME AND QMGRNAME
         MVC   MQMD_FORMAT,MQFMT_IMS         SET MQMD FORMAT NAME
         MVC   MQMD_LENGTH(8,R6),MQFMT_IMS_VAR_STRING
*                                            SET REPLYTO FORMAT NAME
         B     EXIT0
*
EXIT0    DS    0H
         LA    R15,0                         SET RC TO OTMA PROCESS
         B     BYEBYE                        AND EXIT
*
EXIT4    DS    0H
         LA    R15,4                         SET RC TO NON-OTMA
         B     BYEBYE                        AND EXIT
*
BYEBYE   DS    0H
         RETURN (14,12),,RC=(15)             RETURN CODE IN R15
         SPACE 2
         REQUATE
         SPACE 2
         CMQA   EQUONLY=NO
         CMQMDA DSECT=YES
         SPACE 2
         END
```

*Figure 34. Sample assembler DRU exit*

# Appendix C. Enabling distributed queuing using CICS ISC

**Important notice**

Distributed queuing using CICS ISC is retained for compatibility with previous releases; there will be no further enhancements to this function. Therefore, you are recommended to use the channel initiator ("non-CICS mover") for distributed queuing.

To enable distributed queuing using CICS ISC (the "CICS mover"), you must do the tasks described in the following sections:
- "Defining MQSeries programs and data sets as CICS resources"
- "Defining the channel definitions" on page 242
- "Defining the CKMQ transient data queue" on page 242
- "Defining MQSeries queues, triggers, and processes" on page 243
- "Defining CICS resources used by distributed queuing" on page 243
- "Setting up communications" on page 243
- "Security considerations for distributed queuing (using CICS ISC)" on page 244

Prerequisites are that you have installed the CICS mover feature, and that the CICS adapter component has already been set up (see "Chapter 5. Setting up the CICS adapter" on page 85).

## Defining MQSeries programs and data sets as CICS resources

As part of installing the CICS adapter, you might already have updated the CICS system definition (CSD) data set. If you have already done this, go to "Defining the channel definitions" on page 242.

The thlqual.SCSQPROC library includes a member called CSQ4D100. This member contains the resource definition online (RDO) statements required for distributed queuing. These RDO statements must be included in the CSD of both the local CICS system and the remote CICS system to be used by the distributed queuing facility.

**Notes:**
1. You might have to customize CSQ4D100; in particular, the definition for the channel definition data set might have to be changed to include a data set name. There is a note at the beginning of CSQ4D100 that explains this.
2. The CSQKCDF file definition must specify a variable record format, that is, RECORDFORMAT(V). You must not change this format.

The group created is called CSQKDQ1. This group can be included in a group LIST so that the definitions are available at CICS startup. A cold start of your CICS system is required. Figure 35 shows an example of JCL that can be used to do this using the CICS DFHCSDUP offline utility.

### Enabling distributed queuing using CICS ISC

```
//CSDLOOKC EXEC PGM=DFHCSDUP,REGION=4096K
//STEPLIB  DD DSN=CICS330.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=your.cics.csd,DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(100,10))
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DSN=MQM.CSQ1.USER(CSQ4D100),DISP=SHR
//         DD *
  ADD GROUP(CSQKDQ1) LIST(yourlist)
/*
```

*Figure 35. Adding the distributed queuing definitions to the CICS CSD.* This JCL sample assumes that the group CSQKDQ1 does not already exist on your CICS system.

## Defining the channel definitions

You must also define the CSQKCDF data set for the channel definitions to be used by the distributed queuing facility. A data set definition is required on both the local and remote CICS systems.

The member CSQ4CHDF of thlqual.SCSQPROC contains the JCL to define the CSQKCDF data set. You must modify the JCL so that the data set high level qualifier and volume attributes conform to the naming conventions at your installation.

When the data set has been defined this DD statement can be added to your CICS startup procedure:

```
//CSQKCDF  DD DSN=thlqual.CDFILE,DISP=SHR
```

*Figure 36. Adding a DD statement to the CICS startup procedure*

Alternatively, you can modify the DSNAME field of the CSQKCDF file definition in the CSQKDQ1 group to contain the data set name. CICS then dynamically allocates the data set, removing the need to modify the CICS startup procedure.

**Notes:**

1. You must not change the supplied values for the RECORDSIZE and KEYS parameters ((400 400) and (20 8) respectively) of the DEFINE CLUSTER functional command in CSQ4CHDF.
2. You should have only one channel definition file for each queue manager. A single CICS system should own the channel definition file; the other CICS systems should define it as a remote file.
3. The channel definitions must be available, via function shipping if necessary, to all CICS regions running distributed queuing programs.

## Defining the CKMQ transient data queue

Messages from the MQSeries distributed queue management facility are normally sent to the system console. However, these can be routed to the CKMQ extra-partition transient data queue. CSQ4DCT2 contains a sample DCT entry for CKMQ, and CSQ4DCT1 contains a sample DCT entry for the corresponding data set information.

These sample DCT entries might already be incorporated with those of the existing DCT as part of installing the CICS adapter (see step 2 on page 88). If you have not done this, follow the instructions in “System definition” on page 87. Then add a DD statement for the CKMQ transient data queue to your CICS startup procedure. For example, `//MQMMSG  DD SYSOUT=*`.

## Defining MQSeries queues, triggers, and processes

You must include the required queue definitions in your MQSeries subsystem. Distributed queuing requires a queue for use with sequence numbers and logical units of work identifiers (LUWID). You must ensure that a queue is available with the name SYSTEM.CHANNEL.SEQNO.

To pass commands to a running channel program, you need to ensure that a channel command queue exists for your system with the name SYSTEM.CHANNEL.COMMAND.

The member CSQ4DISQ in the thlqual.SCSQPROC library contains the queue definitions required for distributed queuing and examples of definitions of your own that you will need. You must customize this sample before you use it, then you can include this member in the CSQINP2 DD concatenation of the MQSeries startup procedure or you can use the COMMAND function in CSQUTIL utility to issue the required DEFINE commands.

## Defining CICS resources used by distributed queuing

The distributed queuing facilities on the local and remote CICS system require the definition of certain CICS resources for communication to be established. Before starting a channel, you must define these resources using the CICS RDO facility:

**ISC LU 6.2 CONNECTION**
This can be one of:
- An LU 6.2 single session terminal
- An LU 6.2 single session connection
- An LU 6.2 parallel session connection

**SESSIONS**
You must define enough sessions to accommodate all the channels that might be active at the same time.

**PROFILE (optional)**
Profile definitions can be created so that channels are allocated a session from a specific mode group.

For information about the definition of these CICS resources, see these books:
- *CICS Intercommunication Guide* for defining CICS ISC links.
- *CICS System Definition Guide* for guidance on implementing ISC in a CICS system.
- *CICS Resource Definition Guide* manual for defining resources to CICS.

## Setting up communications

For information on this, and all other aspects of distributed queuing using CICS ISC, see the *MQSeries Intercommunication* manual.

## Security considerations for distributed queuing (using CICS ISC)

This section discusses security considerations for the “CICS mover”.

When defining and starting channels for distributed queuing, the transactions used require access to certain MQSeries for OS/390 and CICS resources. The list below shows the transactions that are used for distributed queuing and the access requirements that might be needed. Security is not a mandatory requirement and these examples are only relevant where you are using resource security.

**CKMC**
This transaction will require RACF UPDATE access to the following resources:
- The CSQKCDF VSAM file in CICS
- The SYSTEM.CHANNEL.SEQNO local queue in MQSeries for OS/390
- The SYSTEM.CHANNEL.COMMAND local queue in MQSeries for OS/390

The CKMC transaction only needs RACF UPDATE access to the above resources under certain conditions:
- For the CSQKCDF file, only when the following functions are performed:
  - CREATE a channel
  - COPY a channel
  - DELETE a channel
  - ALTER a channel
- For the SYSTEM.CHANNEL.SEQNO local queue, only when the following functions are performed:
  - RESYNC a channel
  - RESET a channel
  - RESOLVE a channel
- For the system.channel.command local queue when requesting stop for a channel.

  All other functions only require RACF READ access.

**CKSG** This transaction will require RACF READ access to the following resources:
- The CSQKCDF VSAM file in CICS

RACF UPDATE access to the following resources:
- The SYSTEM.CHANNEL.SEQNO local queue in MQSeries for OS/390
- The SYSTEM.CHANNEL.COMMAND local queue in MQSeries for OS/390
- The dead-letter queue (see “Dead-letter queue security” on page 155 for information about how to achieve this)

and RACF ALTER access to the following resources:
- The transmission queue specified in the channel definition in MQSeries for OS/390

**CKSV** This transaction will require RACF READ access to the following resources:

- The CSQKCDF VSAM file in MQSeries for OS/390

RACF UPDATE access to the following resources:

- The SYSTEM.CHANNEL.SEQNO local queue in MQSeries for OS/390
- The SYSTEM.CHANNEL.COMMAND local queue in MQSeries for OS/390
- The dead-letter queue (see "Dead-letter queue security" on page 155 for information about how to achieve this)

and RACF ALTER access to the following resources:

- The transmission queue specified in the channel definition in MQSeries for OS/390

**CKRQ**

This transaction will require RACF READ access to the following resources:

- The CSQKCDF VSAM file in CICS

and RACF UPDATE access to the following resources:

- The SYSTEM.CHANNEL.SEQNO local queue in MQSeries for OS/390
- In MQSeries for OS/390, either
  - The object name passed in the *RemoteQName* field of the MQXQH structure, or
  - The transmission queue representing the remote queue manager, if the value in the *RemoteQMgrName* field of the MQXQH structure does not match the local queue manager name.
- In MQSeries for OS/390 the SYSTEM.CHANNEL.COMMAND local queue
- The dead-letter queue (see "Dead-letter queue security" on page 155 for information about how to achieve this)

**CKRC** This transaction will require RACF READ access to the following resources:

- The CSQKCDF VSAM file in CICS

and RACF UPDATE access to the following resources:

- The SYSTEM.CHANNEL.SEQNO local queue in MQSeries for OS/390
- The SYSTEM.CHANNEL.COMMAND local queue
- In MQSeries for OS/390, either
  - The object name passed in the *RemoteQName* field of the MQXQH structure, or
  - The transmission queue representing the remote queue manager, if the value in the *RemoteQmgrName* field of the MQXQH structure does not match the local queue manager name
- The dead-letter queue (see "Dead-letter queue security" on page 155 for information about how to achieve this)

## Enabling distributed queuing using CICS ISC

# Appendix D. Notices

## Notices

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

| | | |
|---|---|---|
| AIX | BookManager | C/370 |
| CICS | DB2 | IBM |
| IMS | IMS/ESA | Language Environment |
| MQSeries | OpenEdition | OS/390 |
| RACF | RAMAC | RMF |
| SupportPac | VTAM | |

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

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

# 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.

**affinity.** An association between objects that have some relationship or dependency upon each other.

**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.

**cluster transmission queue.** A transmission queue that transmits all messages from a queue manager to any other queue manager that is in the same cluster. The queue is called SYSTEM.CLUSTER.TRANSMIT.QUEUE.

**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 or MQCONNX call, or automatically by the MQOPEN call.

**connection handle.** The identifier or token by which a program accesses the queue manager to which it is connected.

**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.

**Cross Systems Coupling Facility (XCF).** Provides the OS/390 coupling services that allow authorized programs in a multisystem environment to communicate with programs on the same or different OS/390 systems.

| **coupling facility.** On OS/390, a special logical partition that provides high-speed caching, list processing, and locking functions in a parallel sysplex.

## D

**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.

**dequeue.** To remove a message from a queue. Contrast with *enqueue*.

**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*.

**dynamic queue.** A local queue created when a program opens a model queue object. See also *permanent dynamic queue* and *temporary dynamic queue*.

## E

**enqueue.** To put a message on a queue. Contrast with *dequeue*.

**environment.** See *application environment*.

**ESM.** External security manager.

**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.

**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*.

## 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.

**global trace.** An MQSeries for OS/390 trace option where the trace data comes from the entire MQSeries for OS/390 subsystem.

| **globally-defined object.** On OS/390, an object whose definition is stored in the shared repository. The object is available to all queue managers in the queue-sharing group. See also *locally-defined object*.

**GTF.** Generalized Trace Facility.

## H

**handle.** See *connection handle* and *object handle*.

**hardened message.** A message that is written to auxiliary (disk) storage so that the message will not be lost in the event of a system failure. See also *persistent message*.

## I

**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*.

| **inbound channel.** A channel that listens for and receives messages from another queue manager. See also *shared inbound channel*.

**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.

**locally-defined object.** On OS/390, an object whose definition is stored on page set zero. The definition can be accessed only by the queue manager that defined it. Also known as a *privately-defined object*.

**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.** In message queuing applications, a communication sent between programs. In system programming, information intended for the terminal operator or system administrator.

**message channel.** In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender at one end and a receiver at the other end) and a communication link. Contrast with *MQI channel*.

**message channel agent (MCA).** A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue. See also *message queue interface*.

**message 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.

**Open Transaction Manager Access (OTMA).** A transaction-based, connectionless client/server protocol. It functions as an interface for host-based communications servers accessing IMS TM applications through the OS/390 Cross Systems Coupling Facility (XCF). OTMA is implemented in an OS/390 sysplex environment. Therefore, the domain of OTMA is restricted to the domain of XCF.

**OTMA.** Open Transaction Manager Access.

| **outbound channel.** A channel that takes messages from a transmission queue and sends them to another queue manager. See also *shared outbound channel*.

**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*.

| **privately-defined object.** In OS/390, an object whose definition is stored on page set zero. The definition can be accessed only by the queue manager that defined it. Also known as a *locally-defined object*.

**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.

## Q

**queue.** An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages—they point to other queues, or can be used as models for dynamic queues.

**queue manager.** A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. An MQSeries object that defines the attributes of a particular queue manager.

**queue 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.

| **queue-sharing group.** In MQSeries for OS/390, a group of queue managers in the same sysplex that can access a single set of object definitions stored in the shared repository, and a single set of shared queues stored in the coupling facility. See also *shared queue*.

**queuing.** See *message queuing*.

**quiesced shutdown.** In MQSeries, a shutdown of a queue manager that allows all connected applications to disconnect. Contrast with *immediate shutdown* and *preemptive shutdown*. 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*.

**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*.

**repository.** A collection of information about the queue managers that are members of a cluster. This information includes queue manager names, their locations, their channels, what queues they host, and so on.

**repository queue manager.** A queue manager that hosts the *full repository* of information about a cluster.

**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.

## S

**SAF.** System Authorization Facility.

**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.

**shared inbound channel.** In MQSeries for OS/390, a channel that was started by a listener using the group port. The channel definition of a shared channel can be stored either on page set zero (private) or in the shared repository (global).

**shared outbound channel.** In MQSeries for OS/390, a channel that moves messages from a shared transmission queue. The channel definition of a shared channel can be stored either on page set zero (private) or in the shared repository (global).

| **shared queue.** In MQSeries for OS/390, a type of local queue. The messages on the queue are stored in the *coupling facility* and can be accessed by one or more queue managers in a *queue-sharing group*. The definition of the queue is stored in the *shared repository*.

| **shared repository.** In MQSeries for OS/390, a shared DB2 database that is used to hold object definitions that have been defined globally.

**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.

**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 initialization table (SIT).** A table containing parameters used by CICS on start up.

## T

**target library high-level qualifier (thlqual).** High-level qualifier for OS/390 target data set names.

**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.

**thlqual.** Target library high-level qualifier.

**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).

**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.

## X

**XCF.** Cross Systems Coupling Facility.

# Bibliography

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

## MQSeries cross-platform publications

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

- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for SINIX and DC/OSx, V2.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Sun Solaris, Intel Platform Edition, V5.1
- MQSeries for Tandem NonStop Kernel, V2.2.0.1
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT, V5.1

The MQSeries cross-platform publications are:

- *MQSeries Brochure*, G511-1908
- *An Introduction to Messaging and Queuing*, GC33-0805
- *MQSeries Intercommunication*, SC33-1872
- *MQSeries Queue Manager Clusters*, SC34-5349
- *MQSeries Clients*, GC33-1632
- *MQSeries System Administration*, SC33-1873
- *MQSeries MQSC Command Reference*, SC33-1369
- *MQSeries Event Monitoring*, SC34-5760
- *MQSeries Programmable System Management*, SC33-1482
- *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390
- *MQSeries Messages*, GC33-1876
- *MQSeries Application Programming Guide*, SC33-0807
- *MQSeries Application Programming Reference*, SC33-1673
- *MQSeries Programming Interfaces Reference Summary*, SX33-6095
- *MQSeries Using C++*, SC33-1877
- *MQSeries Using Java™*, SC34-5456
- *MQSeries Application Messaging Interface*, SC34-5604

## MQSeries platform-specific publications

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

**MQSeries for AIX, V5.1**

*MQSeries for AIX Quick Beginnings*, GC33-1867

**MQSeries for AS/400, V5.1**

*MQSeries for AS/400® Quick Beginnings*, GC34-5557

*MQSeries for AS/400 System Administration*, SC34-5558

*MQSeries for AS/400 Application Programming Reference (ILE RPG)*, SC34-5559

**MQSeries for AT&T GIS UNIX V2.2**

*MQSeries for AT&T GIS UNIX® System Management Guide*, SC33-1642

**MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1**

*MQSeries for Digital OpenVMS System Management Guide*, GC33-1791

**MQSeries for Compaq Tru64 UNIX, V5.1**

*MQSeries for Compaq Tru64 UNIX Quick Beginnings*, GC34-5684

**MQSeries for HP-UX, V5.1**

*MQSeries for HP-UX Quick Beginnings*, GC33-1869

**MQSeries for OS/2 Warp, V5.1**

*MQSeries for OS/2 Warp Quick Beginnings*, GC33-1868

**MQSeries for OS/390, V5.2**

*MQSeries for OS/390 Concepts and Planning Guide*, GC34-5650

*MQSeries for OS/390 System Setup Guide*, SC34-5651

*MQSeries for OS/390 System Administration Guide*, SC34-5652

*MQSeries for OS/390 Problem Determination Guide*, GC34-5892

*MQSeries for OS/390 Messages and Codes*, GC34-5891

*MQSeries for OS/390 Licensed Program Specifications*, GC34-5893

*MQSeries for OS/390 Program Directory*

**MQSeries link for R/3 Version 1.2**

*MQSeries link for R/3 User's Guide*, GC33-1934

**MQSeries for SINIX and DC/OSx, V2.2**

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

**MQSeries for Sun Solaris, V5.1**

*MQSeries for Sun Solaris Quick Beginnings*, GC33-1870

**MQSeries for Sun Solaris, Intel Platform Edition, V5.1**

*MQSeries for Sun Solaris, Intel Platform Edition Quick Beginnings*, GC34-5851

**MQSeries for Tandem NonStop Kernel, V2.2.0.1**

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

**MQSeries for VSE/ESA V2.1**

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

*MQSeries for VSE/ESA™ System Management Guide*, GC34-5364

**MQSeries for Windows V2.0**

*MQSeries for Windows User's Guide*, GC33-1822

**MQSeries for Windows V2.1**

*MQSeries for Windows User's Guide*, GC33-1965

**MQSeries for Windows NT, V5.1**

*MQSeries for Windows NT Quick Beginnings*, GC34-5389

*MQSeries for Windows NT® Using the Component Object Model Interface*, SC34-5387

*MQSeries LotusScript Extension*, SC34-5404

## Softcopy books

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

## HTML format

Relevant MQSeries documentation is provided in HTML format with these MQSeries products:
- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1 (compiled HTML)
- MQSeries link for R/3 V1.2

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

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

## Portable Document Format (PDF)

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

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

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

PDF versions of relevant MQSeries books are supplied with these MQSeries products:
- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V5.2
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1
- MQSeries link for R/3 V1.2

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

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

## BookManager® format

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

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

## PostScript format

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

## Windows Help format

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

## MQSeries information available on the Internet

The MQSeries product family Web site is at:

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

By following links from this Web site you can:

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

## Related publications

For information about other products that are referred to in this book, see the following books:

### OS/390

- *MVS IPCS Customization*, GC28-1755
- *MVS Initialization and Tuning Reference*, SC28-1752
- *MVS Routing and Descriptor Codes*, GC28-1778
- *MVS System Commands* GC28-1781
- *MVS System Management Facilities (SMF)*, GC28-1783
- *OpenEdition Planning*, SC28-1890
- *MVS Resource Measurement Facility User's Guide*, SC28-1916
- *MVS Planning: APPC/MVS Management*, GC28-1807

### CICS Transaction Server for OS/390

- *System Definition Guide*, SC33-1682
- *Customization Guide*, SC33-1683
- *Resource Definition Guide*, SC33-1684
- *Operations and Utilities Guide*, SC33-1685
- *Intercommunication Guide*, SC33-1695
- *Performance Guide*, SC33-1699
- *RACF Security Guide*, SC33-1701

### CICS for MVS/ESA Version 4

- *System Definition Guide*, SC33-1164
- *Customization Guide*, SC33-1165
- *Resource Definition Guide*, SC33-1166
- *Operations and Utilities Guide*, SC33-1167
- *Intercommunication Guide*, SC33-1181
- *Performance Guide*, SC33-1183
- *CICS-RACF Security Guide*, SC33-1185

### IMS

- *Customization Guide*, SC26-8020

### Security Server

- *Security Server (RACF) Security Administrator's Guide*, SC28-1915
- *Security Server (RACF) Auditor's Guide*, SC28-1916
- *Security Server (RACF) System Programmer's Guide*, SC28-1913
- *Security Server (RACF) External Security Interface (RACROUTE) Macro Reference*, GC28-1922

## Other products

| • *DB2 for OS/390 Administration Guide*, SC26-8957
- *ISPF Dialog Developer's Guide and Reference*, SC28-1273
- *TCP/IP OpenEdition: Planning and Release Guide*, SC31-8303
- *Performance Reporter for OS/390 Administration Guide*, SH19-6816-03
- *APPC Security: MVS/ESA, CICS/ESA, and OS/2*, GG24-3960 (redbook)

# Index

# Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

**To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.**

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address / telephone number / fax number / network ID.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC34-5651-00