MQSeries® for VSE/ESA

System Management Guide

Version 2 Release 1 Modification 2

GC34-5364-04

MQSeries® for VSE/ESA

System Management Guide

Version 2 Release 1 Modification 2

GC34-5364-04

Note!
Before using this information and the product it supports, be sure to read the general information under Appendix I, "Notices," on page 511.

Fifth edition (January 2004)

This edition applies to the following product:
- MQSeries for VSE/ESA Version 2 Release 1 Modification 2

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

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

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

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

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

© Copyright International Business Machines Corporation 1995, 2004. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures vii

Tables ix

About this book xi
Who this book is for xi
What you need to know to understand this book . . xi
How to use this book xi

Summary of changes xiii
Changes in this edition (GC34-5364-04) xiii
Changes in GC34–5364–03 xiv
Changes in GC34–5364–02 xvi
Changes in GC34–5364–01 xvii

Chapter 1. Introduction 1
MQSeries and message queuing 1
Time-independent applications 1
Message-driven processing 1
Messages and queues 1
What messages are 1
What queues are 2
Objects 3
Object names 3
Managing objects 3
MQSeries queue managers. 4
MQSeries queues 4
Channels 5
Clients and servers 6
MQSeries applications in a client-server environment 6
MQSeries and CICS 6

Chapter 2. Installation 7
Contents of the library tape 7
Prerequisites 8
Program number 8
Hardware requirements. 8
Software requirements 8
Features 8
Connectivity 9
Compilers supported for MQSeries for VSE/ESA applications. 9
Delivery 9
Installing MQSeries for VSE/ESA – all users. . . . 9
Installation checkpoint (MQSeries installation) . . 10
Procedures for new users 11
Allocate and initialize the required MQSeries files 11
Installing security 12
Changing the MQER TDQ definition 13
Changing the MQXP TDQ definition 14
Changing the MQIE TDQ definition 15
Other considerations for installing security . . . 16
Preparing CICS for MQSeries 16
Modify CICS start-up deck 17
Recovery and restart 18
Uppercase translation 18
Installation checkpoint (CICS) 18
Starting MQSeries 18
MQSeries initialization 19
Checking MQ is active. 22
MQSeries installation verification test. 23
Local queue verification test 23
Installation checkpoint (installation verification test) 26
Remote queue verification test 27
Post installation verification test CICS modifications 28
Migration procedures for existing users 28

Chapter 3. Configuring network communications 31
MQSeries system definitions required for ACF/VTAM 31
Definitions in CICS for LU 6.2 connections . . . 31
Connection definition 33
Session definition 34
MQSeries for VSE/ESA configuration guidelines . . 35
Queue manager configuration guidelines . . . 35
Channel configuration guidelines 36
Channel exits. 43
Channel security exits 43
Channel send and receive exits 44
Channel message exits. 46
Configuring channel exits 47
Writing and compiling channel-exit programs . . 50
Exit programs in CICS. 51
Channel-exit calls and data structures 51
Channel exit sample 55
Adopt MCA 55
Adopt MCA parameters 56
Bullet-proof channels 57
Bullet-proof channel parameters 58

Chapter 4. System operation 61
MQSeries master terminal displays 61
General panel layout 62
MQSeries master terminal (MQMT) – main menu . 63
Master Terminal transactions 64
Operator screen action keys 65
Configuration functions 65
Global system definition 66
Guidelines for configuring queue managers . . 66
Configuring the queue manager 67
Backing up the configuration file after creating the queue manager 75
Queue definitions 76
Creating local queues 77
Create remote queue 83
Create alias queue 84

© Copyright IBM Corp. 1995, 2004

Create alias queue manager 85
Create alias reply queue 86
Modifying and deleting queue definitions . . . 86
Channel definitions 88
Modifying and deleting channel definitions. . . . 91
Selecting an existing channel definition 91
Modifying an existing channel definition . . . 92
Deleting an existing channel definition 92
Setting channel SSL parameters 92
Setting channel exit parameters 94
Code page definitions 96
Create a user-defined code page 97
Modifying and deleting user-defined code pages . . 98
Modifying an existing code page definition . . . 99
Deleting an existing code page definition. . . . 99
Global system definition display 99
Queue definition display 100
Channel definition display 100
Code page definition display 100
Operations functions 101
Start/Stop queue 101
Open / Close channel 104
Reset message sequence number 105
Initialization of system 106
Queue maintenance 107
Monitoring functions 109
Monitor queues. 110
Monitor channel 112
Browse function 113
Administration via a web browser 114
CICS Web Support 115
CWS MQSeries modules. 115
Using CWS with MQSeries 116
Communications process 117
Message expiry 118
Instrumentation events 120
Queue manager events 121
Channel events 123
Performance events 123
Enabling and disabling events 126
Event queues 128
Format of event messages 129
Event messages. 129
Viewing error logs. 130

Chapter 5. Utilities and interfaces. . . 131
System Administration Control Interface 131
Transactional interface (MQCL) 131
Programmable interface (MQPCMD) 132
Background batch modules 135
MQPUTIL program 135
Using the batch interface 136
Batch interface identifier. 137
Batch interface auto-start 138
Starting the batch interface 138
Stopping the batch interface 138
How to use the batch interface 138
Data integrity 139
Verifying the batch interface 139
Restrictions on using the batch interface . . . 140
VSAM file maintenance 140
Delete all function 140
MQPREORG function 141
Multiple queues sharing a VSAM cluster . . . 141
Reorganizing queue files 142

Chapter 6. Problem determination . . 145
MQSeries setup and local queue operation . . . 145
Has MQSeries run successfully before? 145
Is local queue operation working? 145
Network problems 146
Investigating SNA problems 146
Investigating TCP/IP problems 147
Investigating SSL problems 148
Does the problem affect specific parts of the network? 148
Applications. 149
Are there any error messages? 149
Are there any return codes explaining the problem? 149
Can you reproduce the problem? 149
Have any changes been made since the last successful run? 149
Has the application run successfully before? . . 150
Using the MQSeries API monitor 151
Other areas of investigation 153
Have you obtained incorrect output? 154
Does the problem occur at specific times of the day? 154
Is the problem intermittent? 154
Have you applied any service updates? . . . 154
Does the problem affect only remote queues? 154
Is your application or MQSeries for VSE/ESA running slowly? 155
Application design considerations 155
Effect of message length 156
Searching for a particular message 156
Queues that contain messages of different lengths 156
Use of the MQPUT1 call. 156
Incorrect output 156
Messages that do not appear on the queue . . 157
Messages that contain unexpected or corrupted information 158
Problems with incorrect output when using distributed queues. 158
System log 159
Dead-letter queues 159
Using MQSeries trace 159
Problem determination with clients 159
Terminating clients 160
Error messages with clients. 160
Problems with SSL enabled channels 160
SSL availability 161
Cipher specification support 161
Client authentication failure 162
General channel failure 162

Chapter 7. Message data conversion 163
Data conversion exit programs 164
Using LE/VSE for conversion 164

Building a conversion exit program 165

Chapter 8. Programmable system management. 167

Introduction to Programmable Command Formats (PCFs). 167
The problem PCF commands solve 167
What PCFs are 167
Preparing MQSeries for PCF 168
System command queue. 168
PCF command server. 169
Using PCFs 170
PCF command messages 170
Responses 172
Authority checking for PCF commands. . . . 174
Definitions of the PCFs 174
Error codes applicable to all commands . . . 175
Change channel 176
Change queue 183
Change Queue Manager. 191
Copy Channel 197
Copy Queue. 200
Create Channel. 202
Create Queue 204
Delete Channel. 205
Delete Queue 206
Escape. 208
Inquire Channel 209
Inquire Channel Names 212
Inquire Queue 214
Inquire Queue Manager 217
Inquire Queue Names 220
Ping Queue Manager. 221
Reset Channel 222
Start Channel 223
Start Channel Listener 224
Stop Channel 225
Data responses to commands 226
Structures used for commands and responses . . 240
MQCFH - PCF header 240
MQCFIN - PCF integer parameter 243
MQCFST - PCF string parameter 244
MQCFIL - PCF integer list parameter 247
MQCFSL - PCF string list parameter 248

Chapter 9. MQSeries commands . . . 253

Rules for using MQSeries commands 253
Issuing MQSeries commands 254
MQSC utility program 254
MQPMQSC sample JCL 255
MQSeries command prerequisites 256
Descriptions of the MQSeries commands 256
MQSeries channel commands 256
MQSeries queue commands 263
MQSeries queue manager commands 273

Chapter 10. Secure Sockets Layer services. 279

Installing the SSL feature 279
Configuring the queue manager for SSL 280
TCP/IP settings 280
SSL parameters. 281
Configuring a channel for SSL. 281
SSL channel parameters 282
Activating SSL services 284

Chapter 11. Security 285

Why you need to protect MQSeries resources . . 285
Implementing MQSeries security 285
Resources you can protect 286
Connection security 286
Queue and message security 286
Command security 287
Command resource security 287
Dataset security 288
Using security classes and resources. 288
Resources 288
Switch resources 289
Protecting MQSeries resources. 290
Resource definitions for connection security . . 290
Resource definitions for queue security. . . . 292
Resource definitions for command security . . 296
Resource definitions for command resource security 300
Security implementation checklist 302

Appendix A. CICS control table definitions. 303

Sample file control table entries 303
Sample destination control table entry 305
Sample JCL file definition for CICS deck 306
Sample JCL to create CICS CSD group 307

Appendix B. Application Programming Reference 313

Structure data types 313
MQDLH – Dead-letter header 313
MQGMO – Get message options 313
MQMD – Message descriptor 314
MQOD – Object descriptor 318
MQPMO – Put message options 319
MQTM – Trigger message 320
MQI calls. 321
MQBACK – Back out changes 321
MQCLOSE – Close object 323
MQCMIT – Commit changes 324
MQCONN – Connect queue manager 325
MQDISC – Disconnect queue manager. . . . 327
MQGET – Get message. 328
MQINQ – Inquire about object attributes . . . 332
MQOPEN – Open object 339
MQPUT – Put message. 343
MQPUT1 – Put one message 346
MQSET – Set object attributes 350
Attributes of MQSeries objects. 355
Reason codes 355

Appendix C. Application Programming Guidance 357

Supporting application programs that use the MQI 357
Sample source code provided 358
Compiling your application program 358
Developing applications in the C and PL/I programming languages. 358
Application design guidelines 358
Syncpoints and triggers 359
Syncpoint Rollback 361
Triggers 361

Appendix D. Sample JCL and programs 365
Sample JCL to process MQPUTIL. 365
Sample program TTPTST2.Z 365
TTPTST2.Z 366
Sample program TTPTST3.Z 396
TTPTST3.Z 397
Sample program MQPECHO.Z 425

Appendix E. Example configuration - MQSeries for VSE/ESA Version 2.1.2 . 451
Configuration parameters for an LU 6.2 connection 451
Configuration worksheet 451
Explanation of terms 453
Establishing an LU 6.2 connection 455
Defining a connection 455
Defining a session 455
Installing the new group definition 455
What next? 455
Establishing a TCP/IP connection 456
MQSeries for VSE/ESA configuration 456
Configuring channels. 456
Defining a local queue 460
Defining a remote queue 460
Defining a SNA LU 6.2 sender channel 461
Defining a SNA LU 6.2 receiver channel . . . 462
Defining a TCP/IP sender channel 463
Defining a TCP/IP receiver channel 463

Appendix F. MQSeries clients 465
Client support 465
Security considerations 465
Client code-page conversion tables 465

Appendix G. System messages. . . . 467
API system messages. 467
MQSeries message definitions 468
MQSeries messages 468
MQSeries message codes 468
Console Messages 496
Batch Interface Console Messages 496
Automatic reorganization console messages . . . 498

Appendix H. Security implementation 499
Before you install 499
External security manager configuration 500
System and application users 500
MQSeries datasets. 501
Protecting transactions 502
Resource ownership 503
Resource protection 503
Batch user permissions 504
Client user permissions 505
Command permissions 505
Command resource permissions 506
Trigger permissions 507
CICS startup 507
Starting MQSeries 507
Stopping MQSeries 508

Appendix I. Notices 511
Copyright license 512
Trademarks 512

Glossary of terms and abbreviations 513

Bibliography. 521
MQSeries cross-platform publications 521
MQSeries platform-specific publications 523
Softcopy books 524
BookManager format 524
Portable Document Format (PDF) 524
Windows Help format 524
MQSeries information available on the Internet . . 524

Index 525

Sending your comments to IBM . . . 531

Figures

1. Default global system definition. 20
2. Master terminal main menu 22
3. TTPTST2 screen 23
4. Monitor queues screen 24
5. Browse Queue Records screen – status written 25
6. Browse Queue Records screen – status deleted 26
7. Definitions in CICS using RDO for parallel session partner LU 32
8. Definitions in CICS for single-session capable partner LU 33
9. Definitions in CICS singles-session capable LU 33
10. Outline MQSeries channel definition 37
11. Outline MQSeries extended queue definition 40
12. Channel Definitions screen 48
13. Channel Exit Settings screen 48
14. Communication Setting, Adopt MCA parameters 56
15. Channel Record, bullet-proof channel parameter 59
16. Display screen relationships 62
17. General panel layout 63
18. Master terminal main menu 64
19. Configuration main menu. 66
20. System queue manager information 68
21. Queue manager communications settings 70
22. Queue manager log and trace settings . . . 72
23. Queue Manager event settings 74
24. Queue main options screen 76
25. Local queue definition 77
26. Local queue extended definition 79
27. Remote queue definition 83
28. Alias queue definition 84
29. Alias queue manager definition 85
30. Alias queue reply definition 86
31. Object list screen 87
32. Channel record 88
33. Channel list 91
34. Channel SSL parameters 93
35. Channel Exit settings 95
36. Data conversion definitions 96
37. User code page definition 97
38. Code page object list screen 99
39. Global system definition display 100
40. Operations main menu 101
41. Start / Stop queue control screen 102
42. Open / Close Channel 104
43. Reset channel message sequence 105
44. Initialization of system 106
45. Maintain Queue Message Records 108
46. Monitor Main Menu 110
47. Monitor queues 110
48. Monitor Queues - detail 112
49. Monitor channel definitions 112
50. Monitor channel definitions - detail 113
51. Browse Queue Records 114
52. Global System Definition. 121
53. Extended definition 127
54. Batch interface identifier 137
55. API monitor 151
56. API monitor - browse 152
57. API monitor - hexadecimal format 152
58. PCF parameters 168
59. System command and reply queues 255
60. Queue manager communication settings 280
61. SSL parameters for a channel 282
62. Queues, messages, and applications 357
63. Test System Programs 3 - start 397
64. Channel configuration panel 463

© Copyright IBM Corp. 1995, 2004

Tables

1. Object Characteristics of Connection 34
2. CEMT I CONN display output. 34
3. CEDA V SESS display parameter settings 34
4. Example queue manager configuration 42
5. Example channel configuration 42
6. Example queue configuration 42
7. Identifying API calls 45
8. MQPUTIL program general syntax 135
9. MQSC special characters. 254
10. Supported SSL cipher specifications 282
11. SSL Peer Attribute types 283
12. Classes used by MQSeries 288
13. Switch Resources 289
14. Access levels for queue security 292
15. Command authority for PCF commands 297
16. Command authority for MQSeries commands 298
17. Command authority for MQMT options 299
18. Command resource authority for PCF commands 300
19. Command resource authority for MQSeries commands 301
20. Command resource authority for MQMT options 301
21. Command resource authority for MQMT options 2.5 and 4.0. 301
22. Configuration worksheet for VSE/ESA using APPC 451
23. Configuration worksheet for MQSeries for VSE/ESA 457

© Copyright IBM Corp. 1995, 2004

About this book

MQSeries for VSE/ESA Version 2.1.2—referred to in this book as MQSeries for VSE/ESA or simply MQSeries, as the context permits—is part of the MQSeries family of products. These products provide application programming services that enable application programs to communicate with each other using *message queues*. This form of communication is referred to as *commercial messaging*. The applications involved can exist on different nodes on a wide variety of machine and operating system types. They use a common application programming interface, called the Message Queuing Interface or MQI, so that programs developed on one platform can be readily transferred to another.

This book describes the system administration aspects of MQSeries for VSE/ESA Version 2.1.2 and the services it provides to support commercial messaging in a VSE/ESA environment. This includes managing the queues that applications use to receive their messages, and ensuring that applications have access to the queues that they require.

Who this book is for

Primarily, this book is for system administrators, and system programmers who manage the configuration and administration tasks for MQSeries. It is also useful to application programmers who must have some understanding of MQSeries administration tasks.

What you need to know to understand this book

To use this book, you should have a good understanding of the VSE/ESA operating system, and utilities associated with it. You do not need to have worked with message queuing products before, but you should have an understanding of the basic concepts of message queuing.

How to use this book

Read Chapter 1, “Introduction,” on page 1 first for an understanding of MQSeries for VSE/ESA.

The body of this book contains:
- Chapter 2, “Installation,” on page 7
- Chapter 3, “Configuring network communications,” on page 31
- Chapter 4, “System operation,” on page 61
- Chapter 5, “Utilities and interfaces,” on page 131
- Chapter 6, “Problem determination,” on page 145
- Chapter 7, “Message data conversion,” on page 163
- Chapter 8, “Programmable system management,” on page 167
- Chapter 9, “MQSeries commands,” on page 253
- Chapter 10, “Secure Sockets Layer services,” on page 279
- Chapter 11, “Security,” on page 285

At the back of the book there are some appendixes giving information (which will be incorporated in the appropriate MQSeries books at the next opportunity) on the following topics:
- Appendix A, “CICS control table definitions,” on page 303

© Copyright IBM Corp. 1995, 2004

- Appendix B, “Application Programming Reference,” on page 313
- Appendix C, “Application Programming Guidance,” on page 357
- Appendix D, “Sample JCL and programs,” on page 365
- Appendix E, “Example configuration - MQSeries for VSE/ESA Version 2.1.2,” on page 451
- Appendix F, “MQSeries clients,” on page 465
- Appendix G, “System messages,” on page 467
- Appendix H, “Security implementation,” on page 499

Summary of changes

This section describes changes to this edition of the *MQSeries for VSE/ESA System Management Guide*.

Changes since the previous edition of the book are marked by vertical lines to the left of the changes.

Changes in this edition (GC34-5364-04)

The changes in this edition of the System Management Guide are updates and additions to describe the new features and improvements associated with MQSeries for VSE/ESA V2.1.2.

In addition to minor changes throughout the manual, including updates to screen images, the major additions and modifications to this edition include:

- **Instrumentation events**. In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. Such an event causes the queue manager or channel instance to put a special message, called an event message, on an event queue.

  MQSeries instrumentation events provide information about errors, warnings, and other significant occurrences in a queue manager. You can, therefore, use these events to monitor the operation of queue managers.

  Section "Instrumentation events" on page 120 has been added to explain instrumentation events.
- **Channel exits**. Channel exit programs are called at defined places in the processing carried out by MQSeries Message Channel Agent (MCA) programs. Some of these exit programs work in complementary pairs. For example, if an exit program is called by the sending MCA to encrypt the messages for transmission, the complementary process must be functioning at the receiving end to reverse the process.

  The different types of channel exit program supported by MQSeries for VSE/ESA include security, send, receive and message exits.

  Section "Channel exits" on page 43 has been added to explain channel exits and their use in MQSeries for VSE/ESA.
- **Adopt MCA**. The Adopt MCA feature is an integral feature of MQSeries channel operation. It exists to solve a problem with Message Channel Agent (MCA) receiver tasks falling into an indefinite wait state following a transport error.

  The Adopt MCA feature allows an administrator to specify that MQSeries should automatically stop an orphaned instance of a channel where it receives a new inbound connection request for that channel.

  The administrator can specify the level of checking performed before an orphaned candidate is adopted based on combinations of the channel name (must always match for adoption), and the machine address. This allows for less rigorous checking in, for example a DHCP TCP environment where the partner machine's address may change frequently. The Adopt MCA feature is applicable to TCP/IP channels only. Section "Adopt MCA" on page 55 has been added to explain the Adopt MCA feature.
- **Bullet-proof channels**. MQSeries channels over TCP/IP are difficult to handle when network failures occur. If the TCP/IP connection is broken when an

© Copyright IBM Corp. 1995, 2004

MQSeries channel is active, it is not at all uncommon for the receiving end of the channel to ″hang″ indefinitely in a TCP/IP receive call.

Rather than entering a potentially indefinite TCP/IP receive call, MQSeries can instead enter a receive call for a finite amount of time. At the end of this time, the queue manager has control to decide whether to receive again or to shut down the channel. The facility to ″wake up″ channels waiting on a receive call has been named ″bullet-proof channels″.

Although it is the receiver MCA that is generally waiting for data from the sender, during normal operation, the sender MCA can be waiting for data from a receiver. In this case, following a communication failure, it is the sender MCA that can remain in an indefinite wait state. Consequently, the bullet-proof channels feature applies to both sender and receiver channels.

Section “Bullet-proof channels” on page 57 has been added to explain the bullet-proof channels feature.

- **MQI API expansion for MQINQ and MQSET**. The MQI application program interface (API) calls MQINQ and MQSET have been expanded to support the new queue manager and queue attributes associated with instrumentation events.

  Sections “MQINQ – Inquire about object attributes” on page 332 and “MQSET – Set object attributes” on page 350 have been expanded to describe this new support.
- **PCF and MQSC expansion**. Programmable Command Format (PCF) and MQSeries Command (MQSC) support has been expanded to support new object attributes introduced by instrumentation events and channel exits.

  The expanded support is described in Chapter 8, “Programmable system management,” on page 167 and Chapter 9, “MQSeries commands,” on page 253.

Changes in GC34–5364–03

The changes in this edition of the System Management Guide are updates and additions to describe the new features and improvements associated with MQSeries for VSE/ESA V2.1.2.

In addition to minor changes throughout the manual, the major additions and modifications to this edition include:

- **Programmable Command Formats (PCF)**. PCFs define command and reply messages that can be exchanged between a program and any queue manager (that supports PCFs) in a network. PCF commands can be used in a systems management application program for administration of MQSeries objects: queue managers, queues and channels. The application can operate from a single point in the network to communicate command and reply information with any queue manager, local or remote, via the local queue manager.

  Each queue manager has an administration queue with a standard queue name and applications can send PCF command messages to that queue. Each queue manager also has a command server to service the command messages from the administration queue. PCF command messages can therefore be processed by any queue manager in the network and the reply data can be returned to an application, using a specified reply queue. PCF commands and reply messages are sent and received using the normal Message Queue Interface (MQI). MQSeries for VSE/ESA 2.1.2 supports and processes PCF messages.

  Chapter 8, “Programmable system management,” on page 167 has been added to describe the PCF feature.

- **MQSeries commands (MQSC)**. MQSeries commands provide a uniform method of issuing human-readable commands on MQSeries platforms. MQSC commands can be used to perform administration tasks, for example defining, altering, or deleting MQSeries objects. These commands are intended for use by system programmers, system administrators, and system operators.

  MQSeries for VSE/ESA 2.1.2 provides a batch utility program that accepts MQSC command input and uses the MQ/VSE batch interface and PCF feature components to translate and process the commands.

  Chapter 9, "MQSeries commands," on page 253 has been added to describe MQSeries commands and the MQSC batch utility program.
- **Command security**. Command security checking is carried out when an MQSeries command is issued to be processed by a local queue manager. Commands are issued from the MQ/VSE system administration online interface, via PCF messages, or from the MQSC command utility.

  The following new sections have been added to describe command security and its implications for MQSeries for VSE/ESA:
  "Command security" on page 287
  "Resource definitions for command security" on page 296
  "Command permissions" on page 505
- **Command resource security**. Some MQSeries commands, for example changing a local queue, involve the manipulation of MQSeries resources. Command resource security involves checks to see if a user is allowed access to a specific resource relative to the command being issued.

  The following new sections have been added to describe command resource security and its implications for MQSeries for VSE/ESA:
  "Command resource security" on page 287
  "Resource definitions for command resource security" on page 300
  "Command resource permissions" on page 506
- **Message expiry**. MQSeries messages can be placed on a queue, specifying an expiry interval. This is a period of time expressed in tenths of a second, set by the application that puts the message. The message becomes eligible to be discarded if it has not been removed from the destination queue before this period of time elapses.

  The value is decremented to reflect the time the message spends on the destination queue, and also on any intermediate transmission queues if the put is to a remote queue. When the message is retrieved by an application using the MQGET call, the Expiry field represents the amount of the original expiry time that still remains.

  A new section, "Message expiry" on page 118, has been added to explain the message expiry feature.
- **Optional logging to console**. During normal MQ operation, messages of differing severity are written to the MQSeries system log. These messages might provide information about the system's operation, or highlight a critical failure of one of the system's components.

  MQSeries for VSE/ESA 2.1.2 provides a facility to optionally send messages written to the system log to the VSE/ESA console. Messages sent to the console can be configured to require an operator response. This way, significant messages can be immediately brought to the attention of operations personnel.

  The section "Queue Manager Log and Trace Settings" on page 72 has been expanded to explain the optional logging to console feature.
- **MQI API expansion for MQINQ and MQSET**. The MQI application program interface (API) includes the MQINQ and MQSET calls. These allow application

programs to request or set information about MQSeries queues. The MQINQ call can also be used to request information about the queue manager.

In MQSeries for VSE/ESA 2.1.2, these MQI calls have been expanded to facilitate the inquiry and setting of nearly all queue attributes, and in the case of MQINQ, also inquire on nearly all queue manager attributes.

Sections "MQINQ – Inquire about object attributes" on page 332 and "MQSET – Set object attributes" on page 350 have been expanded to detail the full list of selectors that are supported.

- **Multiple concurrent batch interfaces**. Unlike MQSeries on other platforms, MQSeries for VSE/ESA is implemented as a CICS subsystem. This means that access to MQSeries objects using the message queue interface (MQI) is restricted to CICS applications. To avoid this limitation, MQSeries for VSE/ESA provides an interface for batch programs.

  Prior to V2.1.2, MQ/VSE allowed the activation of only one batch interface per VSE/ESA host. Each interface can manage multiple concurrent batch connections. V2.1.2 allows the activation of multiple concurrent interfaces. This means each queue manager, running in its own CICS region, can now have an active batch interface.

  Section "Using the batch interface" on page 136 has been expanded to explain the multiple concurrent batch interface feature.
- **CICS Web Support**. CICS Web Support is a collection of CICS TS resources supporting direct access to CICS transaction processing services from web browsers. These resources provide tools to generate HTML source from 3270 map-based applications, and interact with CICS programs from a Web browser.

  MQSeries for VSE/ESA, as a CICS subsystem, provides a set of 3270 map-based programs for system administration. V2.1.2 provides versions of these programs, and generated HTML source, so that MQ/VSE administration can be managed from a web browser using CICS Web Support.

  A new section, "Administration via a web browser" on page 114, has been added to explain CICS Web Support and administration of MQSeries from a web browser.
- MQSeries for VSE/ESA diagnostic and error messages are now mixed-case. These message have been reproduced in their mixed-case format in Appendix G, "System messages," on page 467.

Changes in GC34–5364–02

The changes in this edition of the System Management Guide are updates to describe the new Secure Sockets Layer (SSL) enabled channels feature and corrections to existing chapter material.

The major additions to this edition include:

- Secure Sockets Layer services. MQSeries can now activate SSL services on a per channel basis. This is possible through the channel definition. Each channel can identify whether or not SSL services are required when a connection is made or accepted, to or from a remote MQ system or MQ client program.

  A new chapter has been added to describe the new SSL enabled channels feature of MQSeries for VSE/ESA. The new chapter describes the configuration and operation of SSL enabled channels. The new chapter is Chapter 10, "Secure Sockets Layer services," on page 279.
- SSL enabled channels problem determination. A new subsection has been added to Chapter 6, "Problem determination," on page 145, to deal with problems with SSL enabled channels. See "Investigating SSL problems" on page 148.

- The MQCMIT and MQBACK MQI calls are now supported by MQSeries for VSE/ESA. A description of these calls and their parameters is provided in "MQI calls" on page 321.

The major modifications to this edition include:

- MQSeries for VSE/ESA prerequisites have been moved from Appendix A to Chapter 2, "Installation," on page 7. See "Prerequisites" on page 8.
- A full list of MQSeries administration transactions by name and function has been provided. See "Master Terminal transactions" on page 64.
- System operation for queue manager parameters has changed to include new configuration screens for the Global System Definition. These are described in "Configuring the queue manager" on page 67.
- System operation for channel definitions has changed to include new configuration screens for the SSL enabled channels feature. These are described in "Setting channel SSL parameters" on page 92.
- Functionality of the MQ/VSE Command-line interface transaction has expanded. A full list of supported function codes and their meaning is provided in "Transactional interface (MQCL)" on page 131.
- MQSeries for VSE/ESA messages now have an appended severity code. All messages in Appendix G, "System messages," on page 467 have been modified to reflect their relevant severity. The list of messages has also been revised to ensure all messages potentially generated by MQ/VSE are given.

Changes in GC34–5364–01

The changes in this edition of the System Management Guide are updates to describe the new function in MQSeries for VSE/ESA Version 2.1.1.

The major additions to the product for this release include:

- Security control for connections, queues, queue managers, and messages. This is described in Chapter 11, "Security," on page 285 and Appendix H, "Security implementation," on page 499.
- Message data conversion. The queue manager supports conversion of a number of built-in formats. For other formats, you can create a data conversion exit program.This is described in Chapter 7, "Message data conversion," on page 163.
- Improved VSAM file reorganization. The VSAM file associated with a selected queue can be reorganized automatically at specified time intervals. This is described in "Creating local queues" on page 77.
- Enhanced batch interface, described in "Using the batch interface" on page 136.

Other additions include:

- Support for Java clients
- Trigger user data support
- TCP/IP domain name support
- TCP/IP performance and recovery improvements

Changes in this edition (GC34-5364-04)

Chapter 1. Introduction

This chapter introduces MQSeries for VSE/ESA from an administrator's perspective, and describes the basic concepts of MQSeries and messaging.

MQSeries and message queuing

MQSeries lets VSE/ESA applications use message queuing to participate in message-driven processing. Applications can communicate across different platforms by using the appropriate message queuing software products. For example, VSE/ESA and MVS/ESA™ applications can communicate through MQSeries for VSE/ESA and MQSeries for OS/390 respectively. The applications are shielded from the mechanics of the underlying communications.

MQSeries products implement a common application programming interface (message queue interface or MQI) whatever platform the applications are run on. This makes it easier to port applications from one platform to another.

The MQI is described in detail in the *MQSeries Application Programming Reference* manual.

Time-independent applications

With message queuing, the exchange of messages between the sending and receiving programs is time independent. This means that the sending and receiving applications are decoupled so that the sender can continue processing without having to wait for the receiver to acknowledge the receipt of the message. In fact, the target application does not even have to be running when the message is sent. It can retrieve the message after it is started.

Message-driven processing

Applications can be automatically started by messages arriving on a queue using a mechanism known as *triggering*. If necessary, the applications can be stopped when the message or messages have been processed.

Messages and queues

Messages and queues are the basic components of a message queuing system.

What messages are

A *message* is a string of bytes that has meaning to the applications that use it. Messages are used for transferring information from one application to another (or to different parts of the same application). The applications can be running on the same platform, or on different platforms.

MQSeries messages have two parts; the *application data* and a *message descriptor*. The content and structure of the application data is defined by the application programs that use them. The message descriptor identifies the message and contains other control information, such as the type of message and the priority assigned to the message by the sending application.

© Copyright IBM Corp. 1995, 2004

The format of the message descriptor is defined by MQSeries for VSE/ESA. For a complete description of the message descriptor, see the *MQSeries Application Programming Reference* manual.

Message lengths

In MQSeries for VSE/ESA, the maximum message length is 4 MB (where 1 MB equals 1 048 576 bytes). In practice, the message length may be limited by:

- The maximum message length defined for the receiving queue.
- The maximum message length defined for the queue manager.
- The maximum message length defined by either the sending or receiving application.
- The amount of storage available for the message.

  This parameter is extremely important for MQSeries for VSE/ESA. The storage will be used from the CICS® partition in which the queue manager is active.

It may take several messages to send all the information that an application requires.

What queues are

A *queue* is a data structure that stores zero or more messages. The messages may be put on the queue by applications or by a queue manager as part of its normal operation.

Each queue belongs to a *queue manager*, which is responsible for maintaining it. The queue manager puts the messages it receives on the appropriate queues.

Applications send and receive messages using MQI calls. For example, one application can put a message on a queue, and another application can retrieve the message from the same queue. The sending application opens the queue for put operations by making an MQOPEN call. Then it issues an MQPUT call to put the message onto that queue. When the receiving application opens the same queue for gets, it can retrieve the message from the queue by issuing an MQGET call.

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

MQSeries for VSE/ESA supports only *predefined queues*, which are those created by an administrator using the appropriate command set, for example, those defined using the MQSeries Master Terminal (MQMT) utility. Predefined queues are permanent; they exist independently of the applications that use them and survive MQSeries for VSE/ESA restarts.

Retrieving messages from queues

In MQSeries for VSE/ESA, suitably authorized applications can retrieve messages from a queue according to these retrieval algorithms:

- First-in-first-out (FIFO).
- A program request for a specific message, identified by a message identifier or correlation identifier.

The MQGET request from the application determines the method used.

Objects

Many of the tasks described in this book involve manipulating MQSeries *objects*. In MQSeries for VSE/ESA, there are three different types of object:
- Queue managers; see "MQSeries queue managers" on page 4.
- Queues; see "MQSeries queues" on page 4.
- Channels; see "Channels" on page 5.

Object names

Each instance of a queue manager is known by its name. This name must be unique within the network of interconnected queue managers, so that one queue manager can unambiguously identify the target queue manager to which any given message should be sent.

For the other types of object, each object has a name associated with it and can be referenced in MQSeries for VSE/ESA by that name. These names must be unique within one queue manager and object type. For example, you can have a queue and a process with the same name, but you cannot have two queues with the same name.

In MQSeries, names can have a maximum of 48 characters, with the exception of *channel names*, which have a maximum of 20 characters.

Managing objects

MQSeries provides commands for creating, altering, displaying, and deleting objects through the panel driven MQSeries Master Terminal (MQMT) system administration transaction; see "MQSeries master terminal (MQMT) – main menu" on page 63 for further details.

Note: Default object definitions are not supplied with MQSeries for VSE/ESA.

You can perform some limited administration, for example, the starting and stopping of queues and channels, by using the MQCL transaction. See Chapter 5, "Utilities and interfaces," on page 131 for further details.

Local and remote administration

Local administration means carrying out administration tasks on any queue managers you have defined on your local system. You can access other systems, for example through TCP/IP, and carry out administration there. In MQSeries, you can consider this as local administration because no channels are involved, that is, the communication is managed by the operating system.

Object attributes

The properties of an object are defined by its attributes. Some you can specify, others you can only view. For example, the maximum message length that a queue can accommodate is defined by its *MaxMsgLength* attribute (see Figure 26 on page 79). You can specify this attribute when you create a queue.

In MQSeries for VSE/ESA, there are four ways of accessing an attribute:
- Using the MQMT transaction, described in "MQSeries master terminal (MQMT) – main menu" on page 63.
- Using the MQINQ function call, described in "MQINQ – Inquire about object attributes" on page 332.
- Using Programmable Command Format (PCF) messages, described in Chapter 8, "Programmable system management," on page 167.

- Using MQSeries Commands (MQSC), described in Chapter 9, "MQSeries commands," on page 253.

MQSeries queue managers

A queue manager provides queuing services to applications, and manages the queues that belong to it. Under MQSeries for VSE/ESA, there can be one queue manager per CICS region. It ensures that:

- Object attributes are changed according to the commands received.
- Special events such as trigger events are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application making the MQPUT call. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a *local queue* to that queue manager. The queue manager to which an application is connected is said to be the local queue manager for that application. For the application, the queues that belong to its local queue manager are local queues.

A *remote queue* is simply a queue that belongs to another queue manager. A *remote queue manager* is any queue manager other than the local queue manager. A remote queue manager exists on a remote machine across the network, or in a different CICS region on the same VSE/ESA host.

MQI calls

A queue manager object may be used in some MQI calls. For example, you can inquire about the attributes of the queue manager object using the MQI call MQINQ.

Note: You cannot put messages on a queue manager object; messages are always put on queue objects, not on queue manager objects.

MQSeries queues

Queues are defined to MQSeries using the appropriate MQMT transaction, or via PCF requests. The transaction specifies the type of queue and its attributes. For example, a local queue object has attributes that specify what happens when applications reference that queue in MQI calls. Examples of attributes are:

- Whether applications can retrieve messages from the queue (GET enabled).
- Whether applications can put messages on the queue (PUT enabled).
- Whether access to the queue is exclusive to one application or shared between applications.
- The maximum number of messages that can be stored on the queue at the same time (maximum queue depth).
- The maximum length of messages that can be put on the queue.

Using queue objects

In MQSeries for VSE/ESA, there are three types of queue object. Each type of object can be manipulated by the product commands and is associated with real queues in different ways:

1. A *local queue* object identifies a local queue belonging to the queue manager to which the application is connected. All queues are local queues in the sense that each queue belongs to a queue manager and, for that queue manager, the queue is a local queue.

2. A *remote queue object* identifies a queue belonging to another queue manager. This queue must be defined as a local queue to that queue manager. The information you specify when you define a remote queue object allows the local queue manager to find the remote queue manager, so that any messages destined for the remote queue go to the correct queue manager.

   You must also define a transmission queue and channels between the queue managers, before applications can send messages to a queue on another queue manager.
3. An *alias queue object* allows applications to access a queue by referring to it indirectly in MQI calls. When an alias queue name is used in an MQI call, the name is resolved to the name of either a local or a remote queue at run time. This allows you to change the queues that applications use without changing the application in any way—you merely change the alias queue definition to reflect the name of the new queue to which the alias resolves.

   An alias queue is not a queue, but an object that you can use to access another queue.

Specific local queues used by MQSeries

MQSeries uses some local queues for specific purposes related to its operation. You ***must*** define them before MQSeries can use them.

Application queues: A queue that is used by an application (through the MQI) is referred to as an *application queue.* This can be a local queue on the queue manager to which an application is linked, or it can be a remote queue that is owned by another queue manager.

Applications can put messages on local or remote queues. However, they can only get messages from a local queue.

Transmission queues: A *transmission queue* temporarily stores messages that are destined for a remote queue manager. You must define at least one transmission queue for each remote queue manager to which the local queue manager is to send messages directly. For information about the use of transmission queues in distributed queuing, see the *MQSeries Intercommunication* book.

Dead-letter queues: A *dead-letter queue* stores messages that cannot be routed to their correct destinations. This occurs when, for example, the destination queue is full. The supplied dead-letter queue is called SYSTEM.DEAD.LETTER.QUEUE. These queues are also referred to as undelivered-message queues on other platforms.

For distributed queuing, you should define a dead-letter queue for each queue manager.

Event queues: In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. Such an event causes the queue manager or channel instance to put a special message, called an event message, on an event queue. Event queue names are configurable as part of the queue manager's Global System Definition.

Channels

Channels are objects that provide a communication path from one queue manager to another. Channels are used in distributed message queuing to move messages from one queue manager to another. They shield applications from the underlying communications protocols. The queue managers may exist on the same, or

different, platforms. For queue managers to communicate with one another, you must define one channel object at the queue manager that is to send messages, and another, complementary one, at the queue manager that is to receive them.

For information on channels and how to use them, see the *MQSeries Intercommunication* book.

Clients and servers

MQSeries for VSE/ESA supports client-server configurations for MQSeries applications, and can act as a server to which all current MQSeries clients can connect.

Note: There is no VSE/ESA™ client.

An *MQSeries client* is a part of the MQSeries product that is installed on a machine to accept MQI calls from applications and pass them to an *MQI server* machine. There they are processed by a queue manager. Typically, the client and server reside on different machines but they can also exist on the same machine.

An *MQI server* is a queue manager that provides queuing services to one or more clients. For VSE/ESA, there is one MQSeries process for each client connection.

All the MQSeries objects, for example queues, exist only on the queue manager machine, that is, on the MQI server machine. A server can support normal local MQSeries applications as well.

The difference between an MQI server and an ordinary queue manager is that a server has a dedicated communications link with each client. For more information about creating channels for clients and servers, see the *MQSeries Intercommunication* book.

MQSeries applications in a client-server environment

When linked to a server, client MQSeries applications can issue MQI calls in the same way as local applications. The client application issues an MQCONN call to connect to a specified queue manager. Any additional MQI calls that specify the connection handle returned from the connect request are then processed by this queue manager. You must link your applications to the appropriate client libraries. See the *MQSeries Application Programming Guide* for further information.

MQSeries and CICS

Note to users

MQSeries for VSE/ESA runs as a CICS task. Consequently, various features of the product are controlled by CICS itself.

These features include security and recovery. If you install MQSeries for VSE/ESA with the security feature, security will be handled by your External Security Manager (ESM).

Chapter 2. Installation

This chapter describes the procedure for installing MQSeries for VSE/ESA. It consists of the following sections:

1. "Contents of the library tape"
2. "Prerequisites" on page 8
3. "Installing MQSeries for VSE/ESA – all users" on page 9
4. "Procedures for new users" on page 11
5. "Starting MQSeries" on page 18
6. "MQSeries installation verification test" on page 23
7. "Post installation verification test CICS modifications" on page 28
8. "Migration procedures for existing users" on page 28

Contents of the library tape

The distribution tape is in standard IBM® MSHP format and may be stacked or non-stacked format depending on how the product is ordered but should be handled in the same way by the VSE install procedures. The tape will contain a sublibrary for "PRD2.MQSERIES".

This sublibrary contains:

- Copy books, used by your CICS applications whenever you intend to call the MQSeries Message Queuing Interface (MQI).
- Object decks, called at linkedit time when you are building your own MQSeries applications (autolink).
- Phases, to provide MQSeries® operation in CICS and Batch.
- Samples having member type Z. Some of these need to be modified for the VSE/POWER JECL statements, as follows:

```
* ** JOB  to   * $$ JOB
* ** LST  to   * $$ LST
* ** SLI  to   * $$ SLI
* ** EOJ  to   * $$ EOJ
```

The samples are:

MQJCONFG.Z Creation of MQSeries configuration file

MQJSETUP.Z Creation of the setup file

MQJQUEUE.Z VSAM cluster definitions for MQSeries queues

MQJMIGR1.Z Migration of old configuration file (step 1)

MQJMIGR2.Z Migration of old configuration file (step 2)

MQJREORG.Z Batch job to reclaim space of deleted records

MQJUTILY.Z Various batch functions

MQJLABEL.Z Label definitions for the CICS start-up job

MQJCSD.Z Define CICS resources into the CICS CSD

MQCICDCT.Z Entry definitions for CICS DCT

MQCICFCT.Z Entry definitions for CICS FCT

© Copyright IBM Corp. 1995, 2004

MQUSERID.Z	Sample assembler to allow a change of user identifier for MCA communications with remote AS/400® systems
MQJCSD24.Z	Define CICS resources into the CICS CSD for CICS TS customers.
MQBICALL.Z	Sample batch interface program that shows how to write an MQI batch program.
MQBISTOP.Z	Sample program to stop the batch interface from a batch partition.
DCHFMT4.Z	Sample data conversion exit program for message data conversion.

Prerequisites

Program number

- 5686-A06 MQSeries for VSE/ESA Version 2 Release 1.1 (Europe, the Middle East and Africa only).

Hardware requirements

- MQSeries Servers:

  Any IBM System/370™ or System/390® machine
  Minimum system memory – normal memory supplied with machine
  Minimum DASD = VSE library requirements + size of queues
  VSE library requirements:
  3380 = 3 cylinders
  3390 = 2 cylinders
  FBA (Fixed Block Architecture) = 4500 blocks

Software requirements

Minimum supported levels are shown. Later levels, if any, will be supported unless otherwise stated. Note that the latest maintenance for these requirements is strongly recommended.

- VSE/ESA 2.5 (5690-VSE)
- CICS/VSE® 2.3 (5686-026) or CICS TS 1.1 (5648-054)
- VTAM® for VSE/ESA 4.2 (5666-363)

  or

  TCP/IP for VSE/ESA 1.4 (5686-A04)
- LE/VSE 1.4.1 Runtime library (5696-067)
- MQSeries clients:

  MQSeries for VSE/ESA supports clients that can be connected using TCP/IP.

Features

The features described in this book are provided with the MQSeries for VSE/ESA product. Some features, however, are enhancements to the product, and are available only after the relevant APAR/PTFs have been applied.

The following list indicates the APAR prerequisites for certain enhancement features:

- MQSeries commands requires PQ71065.
- Instrumentation events requires PQ75790.
- Channel exits requires PQ79855.
- Adopt MCA requires PQ82520.

- Bullet-proof channels requires PQ82520.

Connectivity

Network protocols supported are SNA LU 6.2 and TCP/IP.
- For SNA connectivity – VTAM for VSE/ESA V4.2
- For TCP/IP connectivity – TCP/IP for VSE/ESA V1.4

Compilers supported for MQSeries for VSE/ESA applications

- Programs can be written using C, COBOL or PL/I
- C programs can use the C for VSE V1.1 compiler (5686-A01)
- COBOL programs can use the COBOL for VSE compiler V1.1 (5686-068)
- PL/I programs can use the PL/I for VSE compiler V1.1 (5696-069)

Delivery

MQSeries for VSE/ESA is available on:
- 3480 cartridge
- 4mm DAT tape

Installing MQSeries for VSE/ESA – all users

To install the product, carry out the following procedure:

1. Decide the name of the :
   - Target sublibrary

     The target sublibrary can be the default supplied, "PRD2.MQSERIES", or a name that you specify.

     If you use the supplied default sublibrary, go to step 2 on page 10.

     If you specify your own library, you must customize the JCL listed in step 1b.
   - VSAM catalog into which the product is to be installed

   a. Create a VSAM user catalog.

      You are recommended to use the Interactive Interface Dialogs (II) to create this catalog. In the following examples, the VSAM catalog named MQMCAT is used, and it is assumed that its label is already defined in the disk label area.

   b. Allocate a VSE library.

      This step is not required if you restore the product into the PRD2 library. However, if you want to install MQSeries in another library, you must create one. You are recommended to use the Interactive Interface dialogs for creating this library, or run the following sample adapted for your environment.

      If you adapt this sample you must modify the sample provided in section 2.b. to use the same sublibrary name.

      - `DEFINE S=lib.sublib` to the your selected name

```
* $$ JOB JNM=MQMSUBL,CLASS=O,DISP=D
// JOB MQMSUBL Define the MQSeries installation library
// DLBL mylib,'l.f.i',yyyy/ddd
// EXTENT ,volume,,,n,m
// EXEC LIBR
DEFINE L=mylib
DEFINE S=mylib.sublib
/*
/&
```

where:
`mylib` is the new library name
`sublib` is the new sublibrary name
`l.f.i` is your local file id
`yyyyyy/ddd`
is the file retention year and day
`volume` is the local disk volume name
`n/m` is the start track and size required

See the IBM VSE System Control Statements documentation for further information about DLBL, EXTENT and LIBR.

2. Restore the MQSeries sublibrary from the library tape. You can do this by either:
   a. Using the Interactive Interface Dialogs, as follows:
      1) From an administrator ICCF signon, select the "Installation" option.
      2) Select "Install Programs – V2 format".
      3) Select "Prepare for installation".

         This presents you with a series of panels and options to identify the tape address and process a job, by scanning the mounted tape and identifying which stacked products are available for installation.

         Monitor the VSE console to see when this job has completed. When it has completed, proceed to the step 2a4.
      4) Select "Install Program(s) from Tape".

         You are presented with a list of products available from the install tape and suggested install sublibraries. You can select either the default install library, "PRD2.MQSERIES", or the name of the customized library you created in Step 1 on page 9.
      5) Select option 1 to proceed with the installation and press function key five (PF5) to create a job to be submitted.

   or

   b. Customizing and processing the following JCL, using the library name from step 1 on page 9.

```
* $$ JOB JNM=MQMTAPE,CLASS=0,DISP=D
// JOB MQMTAPE Restore MQSeries from tape
// ASSGN SYS006,cuu
// MTC REW,SYS006
// EXEC MSHP,SIZE=1M
INSTALL PRODUCT FROMTAPE ID='MQSERIES...2.1.2' -
PROD INTO=lib.sublib
/*
/&
* $$ EOJ
```

Where:

cuu	Is the tape drive address
lib.sublib	Is the sublibrary into which the product is to be installed, for example, PRD2.MQSERIES

Installation checkpoint (MQSeries installation)

You should now have correctly installed the MQSeries sublibrary. This can be verified using a VSE Librarian job to inspect the contents of the library.

The MQSeries phases, objects, and sample jobs are visible.

Note: If the MQSeries product has not installed correctly, check through the preceding instructions to ensure that they all completed correctly.

If you are a new user, see "Procedures for new users." If you are migrating to MQSeries for VSE/ESA V2.1.2 from an earlier release, see "Migration procedures for existing users" on page 28.

Procedures for new users

The following steps describe how to
- Allocate and initialize the required MQSeries files
- Customize your CICS system to utilize the MQSeries facilities

The samples for the following jobs can be found in the installation library you selected, or "PRD2.MQSERIES".

Allocate and initialize the required MQSeries files

You must now run the jobs to:
- Create the setup file
- Create the MQSeries configuration file
- Create cluster definitions for MQSeries queues

Note to users

The sample JCL jobs **must** be modified and customized to refer to your own volume identifiers and catalog names.

This should be done by your VSE systems programmer.

MQJSETUP.Z

Allocate a VSAM ESDS, MQFSSET, which is needed to populate the MQSeries configuration file with text and help messages at initialization time.

Note: Review the section "Installing security" on page 12 before running this sample JCL.

MQJCONFG.Z

Allocates the MQSeries (CICS) subsystem configuration file. For this VSAM KSDS file, each record is a fixed length of approximately 2 KB.

To estimate the space you require, allocate one record, consisting of one cylinder for normal operation, for each MQSeries channel and queue.

MQJQUEUE.Z

Allocates and initializes the MQSeries message queue files. For these VSAM KSDS files, each record is of varying length, depending upon the size of the user data area. A message queue file is required for each queue defined to the MQSeries (CICS) subsystem.

To estimate the space required for each message queue, use the following guidelines:
- Each message queue file contains one header record for each local queue.
- One record is written for each user message.

- Each record is of variable length and consists of a header of 740 bytes plus the actual variable-length user data area.
- This job allocates the following system queue files:
  MQSERIES.MQFERR – Dead letter queue file
  MQSERIES.MQFLOG – Error log queue file
  MQSERIES.MQFMON – Monitor queue file
  MQSERIES.MQFREOR – Automatic VSAM reorganization file

  and optionally the following files:
  MQSERIES.MQFACMD – Admin command file
  MQSERIES.MQFARPY – Admin reply file
  MQSERIES.MQFIEQE – Queue manager events file
  MQSERIES.MQFIECE – Channel events file
  MQSERIES.MQFIEPE – Performance events file

  The following files are sample definitions for user message queues:
  MQSERIES.MQFI001
  MQSERIES.MQFO001
  MQSERIES.MQFI002
  MQSERIES.MQFO002
  MQSERIES.MQFI003
  MQSERIES.MQFO003

You are strongly recommended to define one local queue in each physical file. If you intend to use the automatic VSAM reorganization feature with a queue, that queue must be the only queue in a physical VSAM file.

Installing security

You can protect your MQSeries subsystem from unauthorized access by activating the MQSeries for VSE/ESA security feature. For full details on the security feature, refer to Chapter 11, “Security,” on page 285.

Before installing security, ensure that your environment includes the following prerequisite systems:

- VSE/ESA 2.5 or above.
- CICS TS 1.1 or above.
- External Security Manager (see below).

  You must have an External Security Manager (ESM) that supports the SAF RACROUTE interface. MQSeries for VSE/ESA is not dependent on any specific ESM; however, your ESM should recognize and support standard RACROUTE macro calls. For more information, contact your ESM vendor.

If you have the correct prerequisites and intend to install MQSeries for VSE/ESA security for your queue manager, you must copy and edit the SYSIN.Z installation file, available in the MQSeries installation library PRD2.MQSERIES. You must also change the MQJSETUP.Z sample JCL file that processes the SYSIN.Z file.

The SYSIN.Z file contains installation and configuration parameters that generally should not be changed. However, the file also contains switches for security, which are set off by default and need to be set on to activate security.

To activate the security feature, proceed as follows:

1. Make a copy of your SYSIN.Z file.

   The file resides in your installation library (default PRD2.MQSERIES). When making the copy, note down the target file and sublibrary names for later use.

It is important that you edit the security switches in the copy rather than the original to ensure that the settings are not overwritten by subsequent maintenance operations.

2. Edit the SYSIN.Z file settings:
   a. Search for keyword QMDEF. This is positioned ahead of a list of queue manager definition defaults which, with the exception of the security defaults, can be changed once your system is installed and configured. The security defaults can only be changed by reinstallation.
   b. Locate the default parameter QM-STATUS-SECURITY. The default value for this parameter is DISABLED. To activate security, change the setting to ENABLED and save the file.

      You will also notice a default parameter for security audit. This is not implemented in MQSeries for VSE/ESA 2.1.2. It is reserved for future product extensions.
3. Update the MQJSETUP.Z sample JCL.

   The MQJSETUP.Z file defines a VSAM ESDS and imports the contents of the SYSIN.Z file. You must change the * $$ SLI card in MQJSETUP.Z to identify your SYSIN.Z copy as follows:

   Change:

   ```
   * $$ SLI MEM=SYSIN.Z,S=PRD2.MQSERIES
   ```

   To:

   ```
   * $$ SLI MEM=sysin.copy,S=your.lib
   ```

Once you have made these changes, you can run the MQJSETUP.Z sample JCL to import the contents of the SYSIN.Z file into a VSAM ESDS. The ESDS is processed by installation transaction MQSU to build your starting MQSeries subsystem configuration. See “Starting MQSeries” on page 18. Security installation is not complete until you run the MQSU transaction.

Changing the MQER TDQ definition

Security installation may also require changes to the MQER transient data queue (TDQ) definition of MQSeries for VSE/ESA. The default definition for this TDQ is shipped in file MQCICDCT.Z (see “Preparing CICS for MQSeries” on page 16).

The MQER TDQ definition requires a trigger transaction to be fired every time an entry is written to the TDQ. The transaction that is started is also called MQER. With CICS TS, this transaction will run as the CICS default user (DFLTUSER) unless the DCT definition identifies a USERID.

For security purposes, the user identified with the MQER transaction must have MQSeries CONNECT authority and UPDATE authority to the SYSTEM.LOG queue. Therefore, you must decide whether to grant these privileges to the CICS default user, or to a special user. For security purposes, we recommended that you identify a special user to run the MQER transaction.

If you intend to grant the appropriate authority to the CICS default user, you do not need to change the MQCICDCT.Z sample file. However, if you intend to identify a special user to run the MQER transaction, you need to perform the following:

1. Create a user with your ESM.
2. Grant CONNECT and UPDATE authority to the user. For details on granting security access to users, refer to Chapter 11, “Security,” on page 285.

3. Copy the MQCICDCT.Z file. We recommend that you copy the MQCICDCT.Z file rather than directly edit the base file. The MQCICDCT.Z file is a source fragment that should be included in the DCT source file for your CICS system.
4. Change the MQER TDQ definition in MQCICDCT.Z as follows:

   Change:

```
MQER     DFHDCT TYPE=INTRA,
                RSL=PUBLIC,
                DESTID=MQER,
                DESTFAC=FILE,
                TRANSID=MQER,
                TRIGLEV=1
```

   To:

```
MQER     DFHDCT TYPE=INTRA,
                RSL=PUBLIC,
                DESTID=MQER,
                DESTFAC=FILE,
                USERID=youruser,
                TRANSID=MQER,
                TRIGLEV=1
```

5. Rebuild your DCT phase. Your CICS system programmer can use the MQCICDCT.Z source fragment to do this.

Changing the MQXP TDQ definition

Similar to changes to the MQER TDQ definition, security installation may also require changes to the MQXP transient data queue (TDQ). The default definition for this TDQ is shipped in file MQCICDCT.Z (see Preparing CICS for MQSeries on page 14).

The MQXP TDQ is used by the MQSeries queue manager to expire messages. To expire messages, the MQXP TDQ defines a trigger transaction that is started by CICS when an expiry request is written to the TDQ by the queue manager. The transaction that is started is also called MQXP. With CICS TS, this transaction will run as the CICS default user (DFLTUSER) unless the DCT definition identifies a USERID.

For security purposes, the user associated with the MQXP transaction must have MQSeries CONNECT authority and UPDATE authority to the any ReplyToQ that might exist in the MQMD data structure of an expiring message. The user must also have UPDATE authority to any VSAM file that can contain expired messages. In other words, the MQXP transaction must be run by a user that has UPDATE authority to most, if not all, local queues.

For this reason, it is not recommended that the MQXP transaction runs with the authority of the CICS default user. Instead, it is recommended that the definition for the MQXP TDQ is changed to identify a USERID with the appropriate authority.

To change the MQXP TDQ definition:

1. Create a user with your ESM.
2. Grant CONNECT and queue UPDATE authority to the user. Also ensure that the user has UPDATE authority to relevant VSAM files. For more information about security access to users, refer to Chapter 11, “Security,” on page 285.

3. Copy the MQCICDCT.Z file. We recommend that you copy the MQCICDCT.Z file rather than directly edit the base file. The MQCICDCT.Z file is a source fragment that should be included in the DCT source file for your CICS system.
4. Change the MQXP TDQ definition in MQCICDCT.Z.

   Change:

   ```
   MQXP DFHDCT TYPE=INTRA,
               RSL=PUBLIC,
               DESTID=MQXP,
               DESTFAC=FILE,
               TRANSID=MQXP,
               TRIGLEV=1
   ```

   To:

   ```
   MQXP DFHDCT TYPE=INTRA,
               RSL=PUBLIC,
               DESTID=MQXP,
               DESTFAC=FILE,
               USERID=youruser,
               TRANSID=MQXP,
               TRIGLEV=1
   ```

5. Rebuild your DCT phase. Your CICS system programmer can use the MQCICDCT.Z source fragment to do this.

Changing the MQIE TDQ definition

Similar to changes to the MQXP TDQ definition, security installation may also require changes to the MQIE transient data queue (TDQ). The default definition for this TDQ is shipped in file MQCICDCT.Z (see “Preparing CICS for MQSeries” on page 16).

The MQIE TDQ is used by the MQSeries queue manager to register Instrumentation Event (IE) requests. An instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. Such an event causes the queue manager or channel instance to put a special message, called an event message, on an event queue. To achieve this, the queue manager places an IE request on the MQIE transient data queue. Such requests are processed by the IE processor transaction, also called MQIE.

For security purposes, the user associated with the MQIE transaction must have MQSeries CONNECT authority and UPDATE authority to the event queues. The event queues are identified by the queue manager’s global system definition. The user must also have UPDATE authority to the VSAM files that host the event queues.

Rather than allowing the MQIE transaction to run as the CICS default user, it is recommended that the definition for the MQIE TDQ is changed to identify a USERID with the appropriate authority.

To change the MQIE TDQ definition:

1. Create a user with your ESM.
2. Grant CONNECT and queue UPDATE authority for each of the event queues to the user. Also ensure that the user has UPDATE authority to relevant VSAM files. For more information about security access to users, refer to Chapter 11, “Security,” on page 285.

3. Copy the MQCICDCT.Z file. It is recommended that you copy the MQCICDCT.Z file rather than directly edit the base file. The MQCICDCT.Z file is a source fragment that should be included in the DCT source file for your CICS system.
4. Change the MQIE TDQ definition in MQCICDCT.Z. Change:

```
MQIE  DFHDCT  TYPE=INTRA,
      RSL=PUBLIC,
      DESTID=MQIE,
      DESTFAC=FILE,
      TRANSID=MQIE,
      TRIGLEV=1
```

   to:

```
MQIE  DFHDCT  TYPE=INTRA,
      RSL=PUBLIC,
      DESTID=MQIE,
      DESTFAC=FILE,
      USERID=youruser,
      TRANSID=MQIE,
      TRIGLEV=1
```

5. Rebuild your DCT phase. Your CICS system programmer can use the MQCICDCT.Z source fragment to do this.

Other considerations for installing security

Other installation steps involve:

1. Activation of security classes.
2. Creation of ESM resources.
3. Creation of users.
4. Assignment of resource permissions to users.

Each of these is covered in detail in Chapter 11, “Security,” on page 285.

Preparing CICS for MQSeries

Various CICS tables and definitions must be created and customized for use by the MQSeries subsystem.

You must define the following:

- CICS resources into the CICS CSD
- Entry definitions for the CICS Destination Control Table
- Entry definitions for the CICS File Control Table

The definitions should be reviewed by your CICS systems programmer.

Use the samples (see Appendix D, “Sample JCL and programs,” on page 365) provided with the product. See Appendix A, “CICS control table definitions,” on page 303 for further information.

To help you install the PCT and PPT CICS definitions, the sample MQJCSD.Z is provided. MQJCSD.Z automatically defines the MQSeries entries required into the CICS Definition Data Set (without using migrated CICS, DFHPPT and DFHPCT tables).

You may need to modify this sample to fit your own environment, because all entries are defined in group "MQM", which is then added to the VSELIST list.

MQJCSD.Z – Define CICS resources into the CICS CSD
Sample code that can be used to create CICS-specific PCT and PPT definitions, which are required by the MQSeries subsystem.

MQJCSD24.Z – Define CICS resources for CICS TS
Sample JCL that can be used to create PCT, PPT, and FCT definitions specific to CICS TS that are required by the MQSeries subsystem.

MQCICFCT.Z – File Control Table (FCT)
The sample code provided can be used for creating CICS definitions for the MQSeries configuration and sample queue files. These definitions may require changing to your site's specific requirements.

Note: If you install under CICS TS, you do not need to create your File Control Table (FCT) definitions with this sample. File definitions are provided in the MQJCSD24.Z sample JCL file.

MQCICDCT.Z – Destination Control Table (DCT)
The MQSeries product requires intrapartition transient data queues (TDQ) MQER, MQXP and MQIE, for the processing of log, message expiry and instrumentation events respectively

Note: If you install the security feature, you may need to make special changes to the MQER, MQXP and MQIE transient data queue definitions. See "Installing security" on page 12 for more details.

Modify CICS start-up deck

For CICS applications to use the MQSeries facilities, you must inform CICS of the MQSeries configuration and workfiles, and the location of the MQSeries for VSE/ESA phases as follows:

- Add the label definitions for the CICS start-up job (MQJLABEL.Z) to your CICS start-up deck, or to the standard label procedures. It contains information about the datasets that MQSeries for VSE/ESA uses.

  This step is not necessary when MQSeries is running in a CICS TS environment. However, if required, label definitions in the CICS TS startup JCL can be used to override MQ VSAM file definitions in the CSD.

  Note to users
  This file **must** be modified and customized to refer to the correct volume identifiers and catalog names.

  This should be done by your VSE systems programmer.

- Add the MQSeries for VSE/ESA subsystem install library defined in "Installing MQSeries for VSE/ESA – all users" on page 9 (default name "PRD2.MQSERIES") to the LIBDEF control statement in your CICS startup deck.
- If you are using TCP/IP for queue manager to queue manager or client connections, you must also ensure the PRD1.BASE (TCP/IP base library) is concatenated ahead of the PRD2.SCEEBASE (LE base library). This will ensure that the TCP/IP runtime is correctly referenced.

  For example:

```
// LIBDEF       *,SEARCH=(PRD2.MQSERIES,  *
                      PRD2.CONFIG,        *
                      PRD1.BASE,          *
                      PRD2.SCEEBASE,      *
                      ...)
```

Recovery and restart

Although MQSeries uses its own recovery and restart logic, it also uses standard CICS file management. When MQSeries is running in a CICS/VSE environment, it is important that all MQ VSAM clusters are defined in the DFHFCT with the LOG parameter set to `YES`. In addition, the CICS logging facility should be activated with `JCT = xx` or `YES` in the DFHSIT.

When MQSeries is running in a CICS TS environment, CSD file definitions for MQ datasets should be defined with RECOVERY(BACKOUTONLY).

If you do not fulfill the above conditions, unpredictable results can occur, such as loss of messages or inaccurate values for message sequence numbers.

CICS journal control table

The CICS journal control table (JCT) can be affected by the queue definitions. If a physical record is larger than the buffer size specified in the JCT, a CICS task abend of "AFCL" occurs.

The provided sample FCT queue definitions specify a maximum record length of 4089 bytes. If large records are written, you should set the BUFSIZE parameter of the CICS DFHJCT to a different value; a BUFSIZE value of 4200 is usually sufficient.

For further information, see the *CICS/VSE Resource Definition (Macro)* manual.

This is reflected in either the MQSeries System Log or the CSMT TD queue when an MQPUT call is processed trying to perform this function.

Uppercase translation

Queue manager, queue and channel names are case sensitive on MQSeries systems. If MQSeries for VSE/ESA sends messages to other MQSeries systems, you must specify UCTRAN = TRANID or UCTRAN = NO in your CICS terminal definitions.

If you do not do this, the names you enter into the MQSeries panels are translated into uppercase, and they may not match the actual names on the target MQSeries system.

Installation checkpoint (CICS)

You have now set up the CICS system, and it is ready to be restarted to update the system configuration and utilize the MQSeries subsystem.

Note: If the CICS system has not been updated correctly, check through the preceding instructions to ensure that they all completed correctly.

Starting MQSeries

The MQ CICS environment requires a cold start. Following the restart, the MQSeries for VSE/ESA configuration file must be initialized and populated before the MQSeries for VSE/ESA subsystem can be used.

You do this with the MQSU transaction. However, you are strongly recommended to ensure that all the MQSeries for VSE/ESA subsystem files are available for access by CICS before running this job.

You do this by issuing the CICS transaction:

```
CEMT  INQUIRE FILE(MQF*)
```

All of the MQSeries for VSE/ESA files defined in "Allocate and initialize the required MQSeries files" on page 11 should be visible, and you should be able to open, close, enable, and disable the files.

If you cannot access these files, refer to your CICS systems programmer and review the steps in "Preparing CICS for MQSeries" on page 16.

If the files are accessible, issue the transaction MQSU. This completes with the message "MQSU – MQSeries Install Completed, NNNN input records read". The number of input records represented by NNNN may change depending on the current maintenance level.

Note that you need only run the MQSU transaction once whenever you install MQSeries for VSE/ESA. This rule applies to initial installations of the product, as well as subsequent installations.

MQSeries initialization

Before you initialize your MQSeries for VSE/ESA system, if you decide to install the security feature, you must carry out a basic security implementation first. For details of how to implement security, refer to Chapter 11, "Security," on page 285.

If you have already implemented security, or you do not wish to install the security feature, you can now initialize your MQSeries for VSE/ESA subsystem as follows:

1. Set up the MQSeries for VSE/ESA environment.

   Run MQSE (Setup Environment).

   The response "MQSE:MQSeries environment setup completed" is displayed, after a few seconds.

2. Specify the queue manager name.

   There can be only one queue manager on each MQSeries for VSE/ESA system and each MQSeries system should have a unique queue manager name. The name is specified using the MQMT System Administration transaction, as follows:

   a. Enter the transaction code MQMT on a CICS terminal.
   b. Select option 1 for the "Configuration" menu.
   c. Select option 1 for the "Global System Definition" update screen.

```
12/09/2002          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQ212
14:15:45                 Global System Definition                    CIC1
MQWMSYS                Queue Manager Information                     A001
Queue Manager . . . . . .: VSE.TS.QM1
Description Line 1. . . . .:
Description Line 2. . . . .:
                       Queue System Values
Maximum Number of Tasks . .: 00000100       System Wait Interval : 00000030
Maximum Concurrent Queues .: 00000100       Max. Recovery Tasks  : 0000
Allow TDQ Write on Errors  : Y    CSMT      Allow Internal Dump  : Y
                       Queue Maximum Values
Maximum Q Depth . . . . . .: 00100000       Maximum Global Locks.: 00001000
Maximum Message Size. . . .: 00004096       Maximum Local Locks .: 00001000
Maximum Single Q Access . .: 00000100
                       Global QUEUE /File Names
Local Code Page . . : 01047
Configuration File. : MQFCNFG
LOG Queue Name. . . : SYSTEM.LOG
Dead Letter Name. . : SYSTEM.DEAD.LETTER.QUEUE
Monitor Queue Name. : SYSTEM.MONITOR

Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF6=Upd  PF9=Comms  PF10=Log   PF11=Event
```

Figure 1. Default global system definition

d. Change the "Queue Manager" field to the name that you are giving to your local queue manager.

e. Press function key six (PF6) to update the configuration.

f. Press function key three (PF3) to quit the screen.

You can leave the other fields unchanged.

3. Define system queues

The System Log is a local queue used to record system diagnostic and error messages. It should be defined before the MQ system is started for the first time.

The name of the system log queue is specified in the queue manager's global system definition (MQMT option 1.5). The default name for the queue is SYSTEM.LOG, however, this can be changed using MQMT option 1.1.

To create the system log queue, carry out the following:

a. Type MQMT at the system prompt.

b. Type 1 on the main menu to select Configuration.

c. Type 2 on the Configuration menu to select queue definitions.

The "Queue Main Options" screen will be displayed.

d. Complete the following fields:

- Object Type L
- Object Name SYSTEM.LOG

e. Press PF5 (Add) to display the "Local Queue Definition" screen.

f. Press PF5 (Add) to display the "Queue Extended Definition" screen and change the default values in the following fields:

- Usage mode N (Normal)
- Physical File Name MQFLOG (file name from FCT)
- Maximum Q Depth 00005000
- Maximum Message Length 00002048

g. Press PF5 (Add) to save the changes.

Once the system log queue has been defined to the queue manager the MQ system can be started. Although it is not immediately necessary, it is recommended that the system dead letter, system monitor, system admin command and system admin reply queues are also defined at this time, repeating steps 3a through 3g above. The names of these queues are configurable using MQMT option 1.1. The default names for these queues, and their default CICS filenames are:

```
SYSTEM.DEAD.LETTER.QUEUE    MQFERR
SYSTEM.MONITOR              MQFMON
SYSTEM.ADMIN.COMMAND.QUEUE  MQFACMD
SYSTEM.ADMIN.REPLY.QUEUE    MQFARPY
```

In addition, if instrumentation events are required, it is recommended that the event queues are defined at this time. The event queues are specified as part of the queue manager's global system definition (MQMT 1.1, PF11). Sample VSAM files for the event queues are provided in file MQJQUEUE.Z, and file definitions for CICS are provided in MQCICFCT for CICS 2.3, and MQJCSD24.Z for CICS TS.

4. Define a local queue.

   You must define some local queues to test the operation of the MQSeries for VSE/ESA subsystem. This task is also carried out by using the MQMT transaction.

   The following definitions allow the installation verification program, TST2, to send messages to `ANYQ`.

   Carry out the following procedure:

   a. Type `MQMT` at the system prompt.
   b. Type `1` on the main menu to select Configuration.
   c. Type `2` on the Configuration menu to select queue definitions. The "Queue Main Options" screen appears.
   d. Complete the following fields:
      - Object Type `L`
      - Object Name `ANYQ`
   e. Press PF5 (Add) to display the "Local Queue Definition" screen.
   f. Press PF5 (Add) to display the "Queue Extended Definition"screen and change the default values in the following fields:
      - Usage mode `N (Normal)`
      - Physical File Name `MQFI001 (file name from FCT)`
      - Maximum Q Depth `00000100`
      - Maximum Message Length `00002048`
   g. Press PF5 (Add) to save the changes.
   h. Press PF2 (Options) to return to the Queue Main Options Screen.
   i. Press PF9 (List) to display a selection screen.
   j. On the selection screen, use the cursor keys to select the queue. Press any character key followed by the `Enter` key.

      A screen displays the queue parameters that you have entered. Check that the correct data has been entered.

5. Initialize the MQSeries for VSE/ESA queue manager. There are two ways of doing this. Either:

   a. Type `MQIT` on a CICS terminal.

The response "MQIT: No channel definitions. Initialization completed" is displayed when the process has completed. This is normal.

or

b. Use the MQSeries for VSE/ESA System Administration transaction (MQMT), as follows:
   1) Issue MQMT to display the main menu panel of MQSeries Administration.
   2) Select 2 - Operation.
   3) Select 4 - Initialization/Shutdown.
   4) Type I in the function field and press function key six (PF6).

   Note: If you carry out the initialization before you perform system setup, you receive the message MQ900000:MQSERIES VSE ENVIRONMENT NOT INITIALIZED.

In the future, you can combine Step 1 on page 19 and Step 5 on page 21 by issuing MQSE with the parameter I to perform the initialization step, as follows:

```
MQSE  I
```

The response "MQSE:MQSeries environment setup and initialized" is displayed when the process has completed.

Checking MQ is active

When you have completed the steps listed in "MQSeries initialization" on page 19, the queue manager is active and you can verify this by typing MQMT on a CICS console, to display Figure 2.

```
12/09/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:45:45           *** Master Terminal Main Menu ***                CIC1
MQWMTP                                                              A001
                         SYSTEM IS ACTIVE

                           1. Configuration

                           2. Operations

                           3. Monitoring

                           4. Browse Queue Records


                   Option:




Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Clear/PF3=Exit                           Enter=Select
```

Figure 2. Master terminal main menu

Figure 2 shows the MQ Master Terminal main menu. Ensure that the "SYSTEM IS ACTIVE" message is displayed.

MQSeries installation verification test

The MQSeries subsystem is now ready for the installation verification procedures.

Stop the MQSeries subsystem, using either the MQST transaction, or the Operations Shutdown menu – MQMT option 2.4, and then reinitialize the MQSeries subsystem (see "MQSeries initialization" on page 19).

To carry out the installation verification test you need:

- One local queue
- The sample transaction TST2
- The program TTPTST2 provided with the product
- Access to two terminals

Local queue verification test

The local queue verification test consists of five steps:

1. Initialize the MQSeries runtime environment.
2. Use the test program TTPTST2 to send a number of messages.
3. Use MQMT to verify that these messages are on the queue.
4. Use the test program TTPTST2 to read the messages.
5. Use MQMT to verify that the messages have been delivered.

Step 1 (initializing the MQSeries runtime environment) is achieved by running transaction MQSE, and either MQIT or MQMT option 2.4. Steps 2 through 5 are achieved as follows:

1. On one terminal, issue the transaction code TST2. This invokes the MQSeries for VSE/ESA test program TTPTST2 and produces the screen in Figure 3.

```
TST2 is a test facility for SENDING / RECEIVING messages
The format of command is as follows:
TST2 XXXX NN QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ

(NOTE: parameters are separated by space(s)).
 XXXX  4-character function code, pad with trailing blank
                     HELP - DISPLAY THIS HELP TEXT
                     PUT  - MQPUT  MESSAGES
                     PUT1 - MQPUT1 MESSAGES
                     PUTR - MQPUT W/ REPLY MESSAGE
                     GET  - MQGET  MESSAGES
                     GETD - MQGET W/ BROWSE & DELETE
                     BOTH - MQPUT FOLLOWED BY MQGET
                     INQ  - INQ ABOUT QUEUE (no count NN)
 NN    2-digit number with leading zero (01 TO 99)
 QQQQ  A 48-character field giving the name of a queue.
 An additional prompt will ask for the name of the reply queue for PUTR option.
```

Figure 3. TTPTST2 screen

2. On a second terminal, start MQMT and use option 3.1 to monitor queue operations. This displays the screen in Figure 4 on page 24.

```
12/09/2002           IBM MQSeries for VSE/ESA Version 2.1.2         TSMQ212
15:02:45                  Monitor Queues                            CIC1
MQWMMOQ                                                             A001
                          QUEUING SYSTEM IS ACTIVE
S QUEUE                        FILE     T INBOUND  OUTBOUND        LR   QDepth
  ANYQ                         MQFI001  N IDLE     IDLE             0        0
  SYSTEM.DEAD.LETTER.QUEUE     MQFERR   N IDLE     IDLE             0        0
  SYSTEM.LOG                   MQFLOG   N IDLE     IDLE             0        4
  SYSTEM.MONITOR               MQFMON   N IDLE     IDLE             0        0









Information displayed.
Enter=Refresh  PF2=Return  PF3=Exit  PF7=Back  PF8=Forward
                              PF9=All   Select or PF10=Detail
```

Figure 4. Monitor queues screen

3. On the first terminal, issue:

   ```
   TST2 PUT 10 ANYQ
   ```

   Note: If you type TST2 without parameters, the HELP screen for using TTPTST2 is displayed.

4. TTPTST2 sends the specified messages addressed to ANYQ.

   You receive the following message on successful completion of the transaction:

   ```
   FULL CYCLE HAS BEEN PERFORMED SUCCESSFULLY
     QUEUE USED - ANYQ
     NUMBER OF MESSAGES PROCESSED -  10
     TOTAL SECONDS ..................... -  hh:mm:ss
   ```

   where:
   - `10` is the number of messages you specified (`nn`).
   - `hh:mm:ss` is the time taken to process `nn` messages.

5. Return to the terminal running the MQMT Monitor Queue process.
6. Press the `Enter` key on this terminal.

   The QDEPTH column for queue ANYQ now equals 10. This is the value specified for `nn` in Step 4.

7. Messages on an MQSeries for VSE/ESA queue can be displayed at any time using the MQMT Browse Queue facility (MQMT option 4). Select this option, enter the queue name in the "Object" field, and press the `Enter` key.

   This displays the screen in Figure 5 on page 25.

```
06/21/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQ21CICS
10:19:19                  Browse Queue Records                      CICS
MQMDISP                     SYSTEM IS ACTIVE                        B001

    Object Name: ANYQ
    QSN Number : 00000001       LR-        0, LW-       10, DD-MQFI001
                    Queue Data Record
Record Status : Written.        PUT date/time  : 20000509102430
Message Size  : 00000200        GET date/time  :
Queue line.
THIS IS A MESSAGE TEXT









Information displayed.
    5686-A06 (C) Copyright IBM Corp. 1998,2002. All Rights Reserved.
 ENTER = Process  PF2 = Main Menu    PF3 = Quit   PF4 = Next    PF5 = Prior
 PF7 = Up     PF8 = Down    PF9=Hex/Char   PF10=Txt/Head
```

Figure 5. Browse Queue Records screen – status written

The queue can then be browsed forwards and backwards using function keys four and five (PF4 and PF5).

Note that in this example, the "Record Status" field is `Written`. This indicates that the message has been placed on the queue but not retrieved.

8. Move to the other terminal.
9. At the CICS prompt, type:

   ```
   TST2 GETD  10 ANYQ
   ```

   Note: If you type TST2 without parameters, the HELP screen for using TTPTST2 is displayed.
10. TTPTST2 reads the specified messages from ANYQ.

    You receive the following message on successful completion of the transaction:

    ```
    FULL CYCLE HAS BEEN PERFORMED SUCCESSFULLY
      QUEUE USED - ANYQ
      NUMBER OF MESSAGES PROCESSED -  10
      TOTAL SECONDS ..................... -  hh:mm:ss
    ```

    where:
    - `10` is the number of messages you specified (`nn`).
    - `hh:mm:ss` is the time taken to process `nn` messages.
11. Return to the terminal, running the MQMT Monitor Queue process.
12. Press the `Enter` key. The Monitor Queue screen still displays ANYQ as the only defined queue.

    Notes:

    1. The QDEPTH number, representing the number of messages on the queue, has decreased to zero.
    2. The total number of messages read from the queue (LR) has increased by the number you read using TTPTST2.

13. Use the MQMT Browse facility to view ANYQ. The "Record Status" field has changed to `Deleted`, and the "GET date/time" field is now completed.

    This indicates that the record has now been retrieved by an application. In this case the test transaction TST2 was used with parameters "TST2 GETD 5 ANYQ".

```
06/21/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQ21CICS
10:19:19                   Browse Queue Records                     CICS
MQMDISP                      SYSTEM IS ACTIVE                       B001

    Object Name: ANYQ
    QSN Number : 00000001       LR-        5, LW-       10, DD-MQFI001
                      Queue Data Record
Record Status : Deleted         PUT date/time  : 20000509102430
Message Size  : 00000200        GET date/time  : 20000509102825
Queue line.
THIS IS A MESSAGE TEXT









Information displayed.
    5686-A06 (C) Copyright IBM Corp. 1998,2002. All Rights Reserved.
 ENTER = Process  PF2 = Main Menu    PF3 = Quit   PF4 = Next    PF5 = Prior
 PF7 = Up     PF8 = Down    PF9=Hex/Char   PF10=Txt/Head
```

Figure 6. Browse Queue Records screen – status deleted

Note that MQSeries for VSE/ESA differs from many other MQSeries platforms, in that when a message is retrieved from a queue it is logically deleted but not physically deleted. The messages are merely flagged as "Deleted".

As a consequence of this technique of flagging messages as "written" and "deleted", messages can have their logical state changed, and where necessary, reprocessed.

You can do this using MQMT option 2.5. However, you are advised to carry out this procedure only when you are familiar with the MQSeries for VSE/ESA system.

You have now completed a local installation verification test demonstrating that two applications can send and receive messages through an MQSeries queue.

Installation checkpoint (installation verification test)

You can now:

- Define local queues.
- Start and stop the queue manager.
- Browse queues using MQMT.
- Monitor the status of queues.
- Run simple MQSeries programs that use local queues.

Note: If the installation verification test has not completed, check through the preceding instructions.

Remote queue verification test

In order to expand this test to include a remote link, you must carry out the following steps:

1. Using the appropriate manufacturer's directions, install the prerequisite hardware and software required to support the selected transport protocol (SNA LU 6.2 or TCP/IP).
2. Define the MQSeries channels that you require. See "Channel definitions" on page 88, and coordinate this task with the remote system administrator.

   You will need to define a Sender channel to send messages from MQSeries for VSE/ESA to a remote MQ system, and a Receiver channel to receiver message from a remote MQ system.

   Sender and receiver channels operate in pairs. They must have the same name, for example, a Sender channel under MQSeries might be called VSE1.TO.NT5, and the Receiver channel on the remote NT system would also be called VSE1.TO.NT5. Sender and Receiver channel pairs must also have matching channel parameter values, that is, matching maximum messages size, batch size and wrap sequence.
3. Configure the transmission queues and remote queues required by MQSeries to communicate over the channel – see "Channel definitions" on page 88.

   For the remote queue verification test, you will need to define a remote queue that identifies a local queue on a remote queue manager, and a transmission queue used to temporarily store messages while they are transmitted to the remote queue manager.

   To test remote queuing to MQSeries for VSE/ESA, you will also need a local queue that can be identified in a remote queue definition on a remote queue manager.

Use transaction TST2 to place test messages on a remote queue. These messages will be temporarily stored in the transmission queue identified in the remote queue definition. The MQSeries for VSE/ESA queue manager will subsequently transmit the message to the remote queue manager. Verify that the test messages arrive successfully on the remote system.

Remote MQ systems will have a utility program similar to the TST2 transaction. On some systems, this program is called 'amqsput'. Use the utility program on the remote system to put messages on the remote queue that points to a local queue on the MQSeries for VSE/ESA system. Verify that the test messages arrive successfully on the local queue.

You have now installed and locally verified MQSeries and you can use the administrative programs and the MQI libraries.

However, before your user applications can effectively use the system for message transmission, you must fully configure the system with your queue definitions.

This last step is the most important part of the installation. The requirements are detailed in:

- Chapter 3, "Configuring network communications," on page 31, which provides the configuration guidelines.

- Chapter 4, “System operation,” on page 61, which describes the MQSeries system administration screens used in the configuration.

Post installation verification test CICS modifications

The MQSeries for VSE/ESA subsystem can be started and stopped automatically as part of the normal CICS startup and shutdown procedures. You do this by adding appropriate entries to the CICS Initialization and Shutdown parameters.

Note to users
You **must** not carry out these steps until you have installed MQSeries for VSE/ESA.

CICS Program List Table Post Initialization (PLTPI)

The MQSeries subsystem requires initialization before applications can start using the queue manager. These steps set up the MQSeries environment and initialize the MQSeries resources.

To start MQSeries automatically, you can add the following programs to the CICS initialization PLT (PLTPI) list:

MQPSENV Set up the MQSeries environment.
MQPSTART Initialize the resources.

For example:

```
DFHPLT TYPE=ENTRY, PROGRAM=MQPSENV DFHPLT TYPE=ENTRY, PROGRAM=MQPSTART
```

Other methods are given in “MQSeries initialization” on page 19.

CICS Program List Table Shut Down (PLTSD)

The MQSeries subsystem should be shutdown correctly before shutting down CICS. This can be done:

- Manually, using transaction MQST
- Automatically, by placing the MQSeries program MQPSTOP in the CICS shutdown PLT before the DFHDELIM statement

  For example:

  ```
  DFHPLT TYPE=ENTRY, PROGRAM=MQPSTOP
  ```

  This ensures that MQSeries ends during the first phase of CICS shutdown.

Migration procedures for existing users

You are strongly recommended to review the section “Installing MQSeries for VSE/ESA – all users” on page 9 before proceeding with the instructions in this section.

Conveniently, migration from MQSeries for VSE/ESA 2.1.0 or 2.1.1 does not require the deletion and re-creation of your VSAM datasets. This means your existing files can be used with their existing data. The only exception to this is the MQSeries configuration file.

To carry out the migration, follow the procedure under “Installing MQSeries for VSE/ESA – all users” on page 9 with the following modifications:

1. Do not run sample job MQJCONFG.Z.

Running MCJCONFG.Z will delete and redefine your MQSeries configuration file, which contains, among other things, your application queue definitions.

Instead, the execution of the MQJSETUP.Z job and transaction MQSU will ensure your configuration is upgraded correctly to 2.1.2. MQJSETUP.Z and MQSU are standard steps during installation.

2. Do not run sample job MQJQUEUE.Z.

   The MQJQUEUE.Z job deletes and redefines your MQ VSAM datasets. You must not run this job if you want to keep your existing queue data.

3. Reset your queue manager definition.

   If you have just installed your V2.1.2 system and you expect it to be configured exactly the same as your V2.1.0 or V2.1.1 system, you may need to review your queue manager definition. On installation, the V2.1.2 queue manager definition will adopt installation defaults. These may not match your V2.1.0 or V2.1.1 system.

   Use MQMT option 1.1 to review and modify your global system definition.

4. You must now relink all your MQSeries for VSE/ESA applications with LIBDEF pointing the 2.1.2 sublib in order to include the new MQSeries for VSE/ESA application programming interface objects. Relinking is required for CICS and batch applications.

When, after completing the full installation process with the above modifications, you start MQSeries for VSE/ESA 2.1.2, you should see your existing queues with their previous data, and your existing channel definitions. This completes the migration process for V2.1.0 and V2.1.1 systems.

Note: If the migration has not completed correctly, check through the preceding instructions.

Chapter 3. Configuring network communications

This chapter describes the steps you perform to configure MQSeries to run on the CICS system and communicate with other MQSeries systems. The chapter assumes that your chosen communications software has been installed and correctly configured on your system.

For ACF/VTAM, using MQSeries should not require any changes to the:
- VTAM parameters
- Definition of CICS systems to VTAM

However, you must define all the LUs that are involved.

For TCP/IP, using MQSeries with the TCP/IP communications protocol requires the installation of TCP/IP for VSE/ESA V1.4 or later.

TCP/IP is shipped as part of the VSE/ESA base product in library PRD1.BASE, and simply requires that you install a product key together with your customer information. For further details refer to the *TCP/IP for VSE/ESA User's Guide*.

MQSeries for VSE/ESA does not have any special TCP/IP installation or configuration requirements.

Note: If TCP/IP is to be used as a transport protocol, the TCP/IP phase library must be added to the LIBDEF statement in the CICS startup JCL before the SCEEBASE library.

This is because SCEEBASE contains a TCP/IP phase stub that handles TCP/IP API calls when TCP/IP is not installed.

This chapter describes how to define connections and sessions for LU 6.2 channel connections, and provides guidelines for configuring the queue manager, channels and queues for effective communications.

MQSeries system definitions required for ACF/VTAM

The local MQSeries for VSE/ESA system has to be informed about remote MQSeries systems with which it will communicate. MQSeries has to be defined to:

- MQSeries on CICS (in the network specific parts of the channel definition)
- CICS itself, in one of the following ways:
  - In a TERMINAL definition
  - In CONNECTION/SESSION definitions
  - Through the CICS AUTOINSTALL facility
- VTAM (either predefined, or by VTAM dynamic resource definition), if you are using SNA LU6.2.

Definitions in CICS for LU 6.2 connections

If the CICS end of an MQSeries channel is to initiate the channel connection (that is, the CICS channel-endpoint is a sender), CICS performs an EXEC CICS ALLOCATE. However, this succeeds only if CICS is:
- A contention winner

© Copyright IBM Corp. 1995, 2004

- Already bound
- Not already allocated

If CICS has no definition of the resource, CICS is incapable of formulating a request to VTAM for session establishment. In these circumstances, CICS AUTOINSTALL is inappropriate – autoinstall is for incoming session establishment requests, not for outgoing ones.

Therefore, for sender channel-endpoints on VSE, a definition of the remote system is required at the CICS level.

If the remote system, at the network level, is capable of supporting parallel sessions (for example, it has independent LU 6.2 capability, or it is another CICS system) and, you intend to configure several channels between the two systems, you should use CONNECTION and SESSIONS definitions.

Typical definitions, using the CICS Resource Definition Online (RDO) transaction, CEDA, are shown in Figure 7.

```
DEFINE GROUP(<group name 1>)
CONNECTION(<remote conn>)
NETNAME(<remote luname>)
ACCESSMETHOD(VTAM)
PROTOCOL(APPC)
SINGLESESS(NO)

DEFINE GROUP(<group name 1>)
SESSIONS(<sess name>)
CONN(<remote conn>)
MODE(<logmode 1>)
MAXIMUM(<max sessions>,<max CICS contention winners>)

INSTALL GROUP(<group name 1>)

ADD GROUP(<group name 1>) LIST(<start-up list>) {AFTER(<group name>)}
```

Figure 7. Definitions in CICS using RDO for parallel session partner LU

If the remote LU is capable of only one session, then it may be defined to CICS as either a single-session connection definition or as a terminal definition (Figure 9 on page 33).

```
DEFINE GROUP(<group name 2>)
CONNECTION(<remote conn>)
NETNAME(<remote luname>)
ACCESSMETHOD(VTAM)
PROTOCOL(APPC)
SINGLESESS(YES)

DEFINE GROUP(<group name 2>)
SESSIONS(<sess name>)
CONN(<remote conn>)
MODE(<logmode 2>)
MAXIMUM(1,1)

INSTALL GROUP(<group name 2>)

ADD GROUP(<group name 2>) LIST(<start-up list>) {AFTER(<group name>)}
```

Figure 8. Definitions in CICS for single-session capable partner LU

```
DEFINE GROUP(<group name 3>)
TERMINAL(<remote conn>)
NETNAME(<remote luname>)
TYPETERM(DFHLU62T)
MODENAME(<logmode 2>)

INSTALL GROUP(<group name 3>)

ADD GROUP(<group name 3>) LIST(<start-up list>) {AFTER(<group name>)}
```

Figure 9. Definitions in CICS singles-session capable LU

The CICS supplied typeterm definition, DFHLU62T, provides a suitable terminal type definition. It exists in group DFHTYPE, which should be installed on your system.

Sample definitions for CICS tables can be found in the sublibrary PRD2.MQSERIES. However, other definitions are specific to your environment and you have to create them manually using the CEDA transaction, or DEFINE commands if using the DFHCSDUP batch program.

The definitions consist of a:
- Connection definition – see "Connection definition"
- Session definition – see "Session definition" on page 34

Connection definition

CICS uses the connection name to identify the other systems. For example, if sessions in VSE1 are to converse with sessions in VSE2 and MVS™, you must define both VSE and MVS connections in each direction.

You must also define all the sessions and terminals involved if you are using SNA LU 6.2.

Type `CEDA DEF CONN GROUP(MQSERIES)` to create connections, and set the fields to the following values:

Table 1. Object Characteristics of Connection

Category	Parameter	Desired Value
	Connection	VSE2
	Group	MQSERIES
Connection Identifiers	Netname	vse2lu62
Connection Properties	ACcessmethod	Vtam
	Protocol	Appc
	Datastream	User
	RECordformat	U
Operational Properties	AUtoconnect	Yes
	INService	Yes
Security	ATtachsec	Local

The settings detailed, together with default values are sufficient for operation. For other parameters, refer to the *CICS/VSE 2.3 Resource Definition (Macro)* manual.

You can also display the connection status by typing `CEMT INQ CONN`, to display:

Table 2. CEMT I CONN display output.

STATUS: RESULTS - OVERTYPE TO MODIFY	
Conn(VSE2) Net(xxxxxxxx)	Ins Acq
Conn(MVS) Net(xxxxxxxx)	Ins Rel

Session definition

Type `CEDA DEF SESSION G(MQSERIES)` to create session names. Enter the values shown in Table 3 to complete the fields.

Table 3. CEDA V SESS display parameter settings

Category	Parameter	Desired Value
	Sessions	VSE1VSE2
	Group	MQSERIES
Session Identifiers	Connection	VSE2
Session Properties	Protocol	Appc
	Maximum	00006,00003
	RECEIVEcount	No
	SENDCount	No
	SENDSize	04096
	RECEIVESize	04096
Operational Properties	Autoconnect	Yes
	Buildchain	Yes
	RELreq	No
	Discreq	No
Recovery	RECOvoption	Sysdefault

The settings detailed, together with default values, are sufficient for operation. For other parameters, refer to the *CICS/VSE 2.3 Resource Definition Guide*.

Note: The DFHSIT Table must have the parameter ISC = YES to make the MQSeries system work.

MQSeries for VSE/ESA configuration guidelines

The following guidelines refer to the MQSeries master terminal (MQMT) administration dialogs. For information about using MQMT, see "MQSeries master terminal (MQMT) – main menu" on page 63.

There are three levels of configuration:
- The queue manager
- The channel
- The queue

Some fields are the same in all three levels, for example, the Maximum Message Size.

Notes:

1. The maximum message size defined in the queue manager configuration must be the largest of all those defined in the channels for this queue manager.
2. The size defined in the channel configuration must be equal to, or greater than, the largest message size that is accessing this channel.
3. Each level of maximum message size configuration utilizes different kinds of resources. Unnecessarily large sizes will consume address space.

Queue manager configuration guidelines

When configuring the queue manager (see "Global system definition" on page 66), use the following guidelines:

Maximum Number of Tasks
The maximum number (integer) of simultaneous connections to the queue manager. Though there is a slight overhead for each unused reservation, there is no harm in setting a large number, for example, 200.

Maximum Concurrent Queues
The maximum number of simultaneous open local queues allowed for the queue manager. You are recommended to set this to a large number, for example, 200.

System Wait Interval
The maximum polling time (in seconds) for the system monitor program after the system starts. A value of thirty seconds is usually sufficient.

Note: The system monitor task remains active until the CICS region is shut down, but exists in a wait state until the task is activated by the expiration of the System Wait Interval or by some specific application interface tasks.

The system monitor task starts up the trigger program and schedules the processes that reclaim resources held by applications that have ended abnormally. If there are too many, the System Wait Interval should be reduced to schedule this cleanup process more frequently.

Maximum Q Depth
The maximum number of active messages allowed by the queue manager for each queue. This value serves as the default Maximum Q Depth value when defining a queue. Any inbound message that causes the queue depth to exceed this size will be rejected as "Queue Full".

If this value is smaller than the Maximum Q Depth specified in the queue definition, it becomes the limiting value for the queue. You should set the value to double the maximum number of messages expected to be queued before any application starts to process them.

Maximum Message Size
The maximum number of characters allowed by the queue manager for each message. This field needs only to be large enough to accommodate the largest message. Setting a higher value than necessary wastes resource.

For example, if you anticipate the largest message to be 10 KB (10,240 bytes) you should set this field to 10240.

Note: Messages are stored in VSAM clusters and large messages can span multiple VSAM records. However, you should avoid spanning multiple clusters wherever possible, because of performance implications.

Where an entire message is stored within a single VSAM record, a message header of 740 bytes, for identification and description, is prefixed to the message.

Where a message is split across multiple records, each subsequent record uses a 56-byte header as a prefix to the data.

Maximum Single Q Access
This field defines the maximum number of MQOPEN calls against any queue handled by this queue manager. A value of 1000 calls is an acceptable value, if the maximum number of opens for each queue in the system is 100.

Maximum Global Locks
The maximum number of entries that the queue manager can use to maintain uncommitted MQPUT or MQGET calls, for each queue in the system, for recovery. A value of 500 is normally used.

Maximum Local Locks
The maximum number of entries that the queue manager can use to maintain uncommitted MQPUT or MQGET calls for each queue and task for recovery. Since an entry of a local lock is deleted once an application issues an explicit SYNCPOINT CICS command to commit updates, the more often an application takes the checkpoint, the fewer the maximum number of local locks needed. You should specify a value greater than the largest message batch size for all the channel records. A value of 20 is usually sufficient.

Channel configuration guidelines

Defining the remote MQSeries system to the local queue manager is described in "Channel definitions" on page 88. However, from the point of view of showing where fields in the various definitions have to correspond, an outline MQSeries channel definition is shown in Figure 10 on page 37.

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
 15:19:47                   Channel Record           DISPLAY         CIC1
 MQMMCHN                                                             A000
 Channel  : VSE7.TO.VSE8
  Desc. . :
  Protocol: T (L/T)     Type : S (Sender/Receiver/svrConn)  Enabled : Y

 Sender
  Remote TCP/IP port . . . . : 01414      LU62 Allocation retry num : 00000000
  Get retry number . . . . . : 00000003   LU62 delay fast (secs). . : 00000000
  Get retry delay (secs) . . : 00000015   LU62 delay slow (secs). . : 00000000
  Convert msgs(Y/N). . . . . : N
  Transmission queue name. . : VSE7.VSE8.XQ5
  TP name. . :

 Sender/Receiver
  Connection : VSE8HOST
  Max Messages per Batch . . : 000050     Message Sequence Wrap . . : 999999
  Max Message Size   . . . . : 0005000    Dead letter store(Y/N)  . : N
  Max Transmission Size  . . : 032766     Split Msg(Y/N)  . . . . . : N
  Max TCP/IP Wait  . . . . . : 000300

 Channel record displayed.
 F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

Figure 10. Outline MQSeries channel definition

When configuring the channel, use the following guidelines:

Protocol
The required transport options for this channel. The options are:
- L – LU 6.2 (SNA)
- T – TCP/IP

Remote TCP/IP port
The port number; relevant for TCP/IP defined channels only.

This field is relevant only for sender channels. Receiver channels are started by the MQSeries listener program which uses the port number configured in the global system definition.

Type Channel type of sender, receiver, or client.

Connection
The channel partner name. This is the CICS connection ID for LU 6.2 channels, or the remote hostname or IP address for TCP/IP channels.

For TCP/IP this field is relevant only for sender channels. Sender channels identify a specific host for communications, whereas receiver channels can accept communications from any host.

Note that for TCP/IP channels, the connection name does not include a port number as it may do on other MQ platforms. The port number is configured as a separate channel parameter.

LU62 Allocation Retry Num
The retry count field represents the number of times an allocation is retried when the conversation has not been established. You should set the retry count at less than 10. If this value is exceeded, the system can be placed under stress.

For receiver channels, this value should be set to zero.

Note: When you configure a new environment, failures occurring more frequently than this can indicate a network problem. You should investigate the problem LU, and its associated resources, to ensure that the session is bound and to establish why the conversation cannot be allocated.

LU62 Delay fast
The time interval, in seconds, that an allocation of conversation is retried for the first cycle of retries. A value of one to five seconds is sufficient for this field, with the longer time being used for a slow environment, for example, a dial-up SDLC.

For receiver channels, this value should be set to zero.

LU62 Delay slow
The time interval, in seconds, that an allocation of conversation is retried for the next cycle of retries, should the first cycle of retries fail. A value between three and 10 seconds is sufficient for this field, with the longer time being used for a slow environment.

For receiver channels, this value should be set to zero.

Get retry number
The number of retries for the MQGET call when the queue is depleted. If a transmission queue is empty, the queue manager retries at the Delay-Time interval before disconnecting the channel or making a request to disconnect the channel.

For receiver channels, this value should be set to zero.

Get retry delay
The time interval, in seconds, between retries. The value of this field may depend on the size of message and the platforms where the LU resides. The optimum value can vary from 1 to 20 seconds.

The longer the delay time specified, the less frequently a channel is reopened. For time-consuming dial-up connections, you are recommended to use a value of 20 seconds.

For receiver channels, this value should be set to zero.

Note: By using a value of zero for the Number of Retries, and a value of "n" seconds for the Delay Time it is possible for you to set a simple disconnection interval similar to that provided on other MQSeries platforms.

Max Messages per Batch
The maximum number of messages in the batch.

Message Sequence Wrap
The message sequence number (MSN) wrap count represents the highest MSN value used on this channel, after which the MSN reverts to one. You are recommended to set this value to 999 999.

Note: The value of the MSN Wrap count must be the same at both the sending and receiving ends of the channel.

Max Transmission Size
The mutually accepted maximum number of characters for each transmission. The minimum value should be equal to the maximum message size expected on this channel, plus 476 bytes for the transmission header.

Max Message Size
The maximum number of bytes for each message that is allowed for this channel.

Convert Msgs
A field that identifies whether message data is converted before it is sent to a remote queue manager. To convert message data, set this field to Y.

Split Msg
A field that identifies whether message data can be split across network transmissions. For example, if the transmission size is 8 KB and message data lengths are up to 30 KB, then the message data must be split across transmissions.To split message data in such situations, set this field to Y.

TP Name
The remote task ID, character only, of the receiver on a remote CICS region or a Transaction Program name on a remote system. This is required by the sender, and since CICS uses four bytes as the transaction identifier, only the first four bytes of the remote task ID are meaningful for CICS to CICS conversation.

This field is not relevant for TCP/IP channels.

Note: VSE converts the name to uppercase, therefore, the corresponding name on the remote system should be defined in uppercase characters.

Max TCP/IP Wait
The maximum number of second that a Message Channel Agent (MCA) should wait to receive TCP/IP data before terminating the connection with an error. See “Bullet-proof channels” on page 57 for more information

Queue configuration guidelines

Defining queues to the local queue manager is described in “Queue definitions” on page 76. Certain parameters in queue definitions are important when configuring network communications. The queue extended definition, shown in Figure 11 on page 40, includes these parameters.

```
11/11/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
 13:00:08               Queue Extended Definition                    CIC1
 MQMMQUE                                                             A005

 Object Name: MQTS.MQ23.XQ1

 General               Maximums                    Events
 Type   . . : Local    Max. Q depth . : 00001000  Service int. event: N
 File name  : MQFO002  Max. msg length: 00004000  Service interval  : 00000000
 Usage  . . : T        Max. Q users . : 00000100  Max. depth event  : N
 Shareable  : Y        Max. gbl locks : 00000200  High depth event  : N
                       Max. lcl locks : 00000200  High depth limit  : 000
                                                  Low depth event . : N
 Triggering                                       Low depth limit . : 000
 Enabled  . : Y        Transaction id.:
 Type . . . : E        Program id . . : MQPSEND
 Max. starts: 0001     Terminal id  . :
 Restart  . : N        Channel name . : MQTS.TO.MQ23
 User data  :
            :

 Requested record displayed.
 PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
             PF9=List  PF10=Queue
```

Figure 11. Outline MQSeries extended queue definition

When configuring the queue (see "Queue definitions" on page 76), use the following guidelines:

File name
The CICS file name, of up to seven characters, used to store messages for this queue. A physical file can hold as many queues as required. A message queue can be logically replenished, if its associated physical file name is changed.

Note: You should not use the MQFREOR file for queue definitions. The MQFREOR file is used by the automatic VSAM reorganization feature and is deleted and redefined during reorganization. Therefore, message data for queues defined in MQFREOR would be lost.

Max. Q depth
The maximum number of records that can remain unread on this queue. Any inbound message that causes the queue depth to exceed this size is rejected as "Queue Full". The minimum value you set should be the maximum number of messages on the queue before the application starts to read and process the queue. In practice, you can set this to 9,999,999.

Max. msg length
The maximum number of characters for each message that this queue allows. If this queue is a transmission queue, the value needs to be sufficiently large to accommodate all messages using this queue as the outbound queue.

Max. Q users
The maximum number of MQOPEN calls that can occur on this queue. You are recommended to set a value of 100 for each queue that is not a transmission queue. For a transmission queue you should add a value of

100 calls, to the base of 100 calls, for each additional target queue that receives messages from this transmission queue. Setting a high value can use too much overhead.

Max. gbl locks
The maximum number of entries that the queue manager uses to maintain committed MQPUT and MQGET calls for this queue for system recovery. If the queue is intended for random message retrieval, rather than sequential processing, then specify a higher value (for example, 1000). For sequential processing, a lower value (for example, 200) should be sufficient.

Max lcl locks
The maximum number of entries that the queue manager uses to maintain uncommitted MQPUT and MQGET calls for this queue for recovery. Since an entry of a local lock is deleted once an application issues an explicit SYNCPOINT CICS command to commit updates, the more often an application takes the checkpoint, the fewer the maximum number of local locks needed. A value similar to the Global Lock Entries setting is recommended.

Trigger Type
"F" is used to generate a trigger when an MQPUT call changes the status of a queue from empty to nonempty. The triggered transaction must have sufficient logic to empty the queue, including messages that may arrive during the process, in a single thread. "E" is used to generate a trigger whenever an MQPUT call occurs and may have as many threads as specified in Max Trigger Starts.

Max. starts
The maximum number of trigger threads that can be activated simultaneously. This field applies to Trigger Type "E" only.

Transaction id
The transaction to be started by the trigger. This field is mutually exclusive with the Program ID. You are recommended to leave this field blank and use a Program ID, for example MQPSEND, unless you require a user transaction.

Once the initial maximum trigger starts is reached then MQSeries for VSE/ESA only checks that the maximum trigger starts are running at every system interval and not when each task completes. If it is important to have a definite number of trigger instances running against a queue, you should use Program ID to identify your trigger program.

Program id
You should use the MQPSEND call on a transmission queue if you require triggering.

Terminal id
You should leave this field blank unless you require a terminal for problem determination purposes.

Channel Name
This field should be left blank except for a transmission queue definition. For a transmission queue definition, this field must identify a channel name.

User Data
This field is for static data that you want to pass to the trigger instance. When a trigger instance is activated, it is passed data in the form of the

MQTM structure (see the CMQTML and CMQTMV copybooks). Data in the User Data field is passed in the MQTM-USERDATA field.

Event The event settings for queues do not affect network communications. For a full description of the event settings, refer to "Local queue extended definition screen" on page 79.

Permitted number of channels

The limit on the number of channels depends upon the availability of system resources. The queue manager can support as many channels and transmission queues as the resource in the system permits.

Example configuration

The following tables give a set of values that can be used to set up your system. See:

- Table 4 for the queue manager
- Table 5 for a channel
- Table 6 for a queue

Table 4. Example queue manager configuration

Parameter	Value	Units
Maximum Number of MQCONN	200	integer
Maximum Open Queue	200	integer
System Wait Interval	30	seconds
Maximum Q Depth	9999999	integer
Maximum Message Size	5000	bytes
Maximum Number of Opens	500	integer
Max Number of Global Locks	500	integer
Max Number of Local Locks	500	integer

Table 5. Example channel configuration

Parameter	Value	Units
Allocation Retries	4	integer
Delay Time-Fast	1	second
Delay Time-Slow	3	seconds
Get Retries	1	integer
Delay Time	10	seconds
Message Sequence Wrap	999999	integer
Maximum Transmission Size	3821	bytes
Maximum Message Size	5000	bytes

Table 6. Example queue configuration

Parameter	Value	Units
Maximum Q Depth	999999	integer
Maximum Message Size	5000	bytes
Maximum Number of Opens	1000 (transmit queue) 100 (other queues)	integer
Max Number of Global Locks	1000 (transmit queue) 100 (other queues)	integer

Table 6. Example queue configuration (continued)

Parameter	Value	Units
Max Number of Local Locks	1000 (transmit queue) 100 (other queues)	integer
Trigger Type	E	character
Maximum Trigger Starts	1	integer
Transaction Id	<blank>	character
Program Id	MQPSEND (transmit queue) user app. (other queues)	character

Channel exits

Channel-exit programs are called at defined places in the processing carried out by MQSeries Message Channel Agent (MCA) programs.

Some of these user-exit programs work in complementary pairs. For example, if a user-exit program is called by the sending MCA to encrypt the messages for transmission, the complementary process must be functioning at the receiving end to reverse the process.

The different types of channel-exit program include:

Security exit
Send exit
Receive exit
Message exit
Message retry exit
Auto-definition exit
Transport retry exit

MQ/VSE does not support message retry, auto-definition or transport retry exits since these exits involve features of MQ that are not supported by MQ/VSE.

Review section “Features” on page 8 for prerequisites for this feature.

Channel security exits

You can use security exit programs to verify that the partner at the other end of a channel is genuine.

Channel security exit programs are called at the following places in an MCA’s processing cycle:

- At MCA initiation and termination.
- Immediately after the initial data negotiation is finished on channel startup. The receiver end of the channel may initiate a security message exchange with the remote end by providing a message to be delivered to the security exit at the remote end. It may also decline to do so. The exit program is re-invoked to process any security message received from the remote end.
- Immediately after the initial data negotiation is finished on channel startup. The sender end of the channel processes a security message received from the remote end, or initiates a security exchange when the remote end cannot. The exit program is re-invoked to process all subsequent security messages that may be received.

Channel send and receive exits

You can use the send and receive exits to perform tasks such as data compression and decompression. In MQ/VSE you can configure a channel for only one send/receive exit program (MQ/VSE does not support exit lists).

Channel send and receive exit programs are called at the following places in an MCA's processing cycle:

- The send and receive exit programs are called for initialization at MCA initiation and for termination at MCA termination.
- The send exit program is invoked at either end of the channel, immediately before a transmission is sent over the link.
- The receive exit program is invoked at either end of the channel, immediately after a transmission has been taken from the link.

There may be many transmissions for one message transfer, and there could be many iterations of the send and receive exit programs before a message reaches the message exit at the receiving end.

The channel send and receive exit programs are passed an agent buffer containing the transmission data as sent or received from the communications link. For send exit programs, the first eight bytes of the buffer are reserved for use by the MCA, and must not be changed. If the program returns a different buffer, then these first eight bytes must exist in the new buffer. The format of data presented to the exit programs is not defined.

A good response code must be returned by send and receive exit programs. Any other response will cause an MCA abnormal end (abend). Since data traffic can continue after an abnormal end (so as to communicate the channel failure to the remote MCA), send and receive exits are automatically suppressed following an bad response code. However, the exit is still called at termination of the channel.

Send and receive exits usually work in pairs. For example a send exit may compress the data and a receive exit decompress it, or a send exit may encrypt the data and a receive exit decrypt it. When you define the appropriate channels, make sure that compatible exit programs are named for both ends of the channel.

Channel send and receive exits may be called for message segments other than for application data, for example, status messages. They are not called during the startup dialog, nor the security check phase.

Although message channels send messages in one direction only, channel-control data flows in both directions, and these exits are available in both directions, also. However, some of the initial channel startup data flows are exempt from processing by any of the exits.

There are circumstances in which send and receive exits could be invoked out of sequence; for example, if you are running a series of exit programs or if you are also running security exits. Then, when the receive exit is first called upon to process data, it may receive data that has not passed through the corresponding send exit. If the receive exit were just to perform the operation, for example decompression, without first checking that it was really required, the results would be unexpected.

You should code your send and receive exits in such a way that the receive exit can check that the data it is receiving has been processed by the corresponding send exit. The recommended way to do this is to code your exit programs so that:

- The send exit sets the value of the ninth byte of data to 0 and shifts all the data along one byte, before performing the operation. (The first eight bytes are reserved for use by the MCA.)
- If the receive exit receives data that has a 0 in byte 9, it knows that the data has come from the send exit. It removes the 0, performs the complementary operation, and shifts the resulting data back by one byte.
- If the receive exit receives data that has something other than 0 in byte 9, it assumes that the send exit has not run, and sends the data back to the caller unchanged.

When using security exits, if the channel is ended by the security exit it is possible that a send exit may be called without the corresponding receive exit. One way to prevent this from being a problem is to code the security exit to set a flag, in MQCD.SecurityUserData or MQCD.SendUserData, for example, when the exit decides to end the channel. Then the send exit should check this field, and process the data only if the flag is not set. This prevents the send exit from unnecessarily altering the data, and thus prevents any conversion errors that could occur if the security exit received altered data.

In the case of MQI channels for clients, byte 10 of the agent buffer identifies the API call in use when the send or receive exit is called. This is useful for identifying which channel flows include user data and may require processing such as encryption or digital signing.

The following table shows the data that appears in byte 10 of the channel flow when an API call is being processed (note that these are not the only values of this byte; there are other reserved values):

Table 7. Identifying API calls

API call	Value of byte 10
MQCONN request[1, 2]	X'81'
MQCONN reply[1, 2]	X'91'
MQDISC request[1]	X'82'
MQDISC reply[1]	X'92'
MQOPEN request	X'83'
MQOPEN reply	X'93'
MQCLOSE request	X'84'
MQCLOSE reply	X'94'
MQGET request[3]	X'85'
MQGET reply[3]	X'95'
MQPUT request[3]	X'86'
MQPUT reply[3]	X'96'
MQPUT1 request[3]	X'87'
MQPUT1 reply[3]	X'97'
MQSET request	X'88'
MQSET reply	X'98'

Table 7. Identifying API calls (continued)

MQINQ request	X'89'
MQINQ reply	X'99'
MQCMIT request	X'8A'
MQCMIT reply	X'9A'
MQBACK request	X'8B'
MQBACK reply	X'9B'

Notes:

1. The connection between the client and server is initiated by the client application using MQCONN. Therefore, for this command in particular, there will be several other network flows. This also applies to MQDISC that terminates the network connection.
2. MQCONNX is treated in the same way as MQCONN for the purposes of the client-server connection.
3. If the message data exceeds the transmission segment size, there may be a large number of network flows per single API call.

Channel message exits

You can use the channel message exit for the following:

- Encryption on the link
- Validation of incoming user IDs
- Substitution of user IDs according to local policy
- Message data conversion
- Journaling
- Reference message handling

In MQ/VSE you can configure a channel for only one message exit program (MQ/VSE does not support exit lists).

Channel message exit programs are called at the following places in an MCA's processing cycle:

- At MCA initiation and termination
- Immediately after a sending MCA has issued an MQGET call
- Before a receiving MCA issues an MQPUT call

The message exit is passed an agent buffer containing the transmission queue header, MQXQH, and the application message text as retrieved from the queue. (The format of MQXQH is given in the MQSeries Application Programming Reference book.) If you use reference messages, that is messages that contain only a header which points to some other object that is to be sent, the message exit recognizes the header, MQRMH. It identifies the object, retrieves it in whatever way is appropriate appends it to the header, and passes it to the MCA for transmission to the receiving MCA. At the receiving MCA, another message exit recognizes that this is a reference message, extracts the object, and passes the header on to the destination queue. See the MQSeries Application Programming Guide for more information about reference messages and some sample message exits that handle them.

Message exits can return the following responses:

- Send the message (GET exit). The message may have been changed by the exit. (This returns MQXCC_OK.)
- Put the message on the queue (PUT exit). The message may have been changed by the exit. (This returns MQXCC_OK.)
- Do not process the message. The message is placed on the dead-letter queue (undelivered message queue) by the MCA.
- Close the channel.
- Bad return code, which causes the MCA to abend.

Message exits are called just once for every complete message transferred, even when the message is split into parts. An exit runs in the same thread as the MCA itself. It also runs inside the same unit of work (UOW) as the MCA because it uses the same connection handle. Therefore, any calls made under syncpoint are committed or backed out by the channel at the end of the batch. For example, one channel message exit program can send notification messages to another and these messages will only be committed to the queue when the batch containing the original message is committed.

Therefore, it is possible to issue syncpoint MQI calls from a channel message exit program.

Configuring channel exits

Channel exits, and their associated exit data, can be configured using:

- Master Terminal transaction (MQMT)
- Programmable Command Formats (PCF)
- MQSeries Commands (MQSC)

Configuration using MQMT

Channel definitions can be created and modified using the master master terminal transaction, MQMT option 1.3, “Channel Definitions”.

The Channel Definitions screen appears as follows:

```
12/17/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
10:41:01                   Channel Record           DISPLAY         CIC1
MQMMCHN                                                             A000
Channel  : VSE7.TO.OS390X
 Desc. . :
 Protocol: T (L/T)     Type : S (Sender/Receiver/svrConn)  Enabled : N

Sender
 Remote TCP/IP port . . . . : 01414      LU62 Allocation retry num : 00000003
 Get retry number . . . . . : 00000003   LU62 delay fast (secs). . : 00000005
 Get retry delay (secs) . . : 00000005   LU62 delay slow (secs). . : 00000001
 Convert msgs(Y/N). . . . . : N
 Transmission queue name. . : AIX2.XQ1
 TP name. . :

Sender/Receiver
 Connection : AIXSERV2
 Max Messages per Batch . . : 000001     Message Sequence Wrap . . : 999999
 Max Message Size . . . . . : 0004096    Dead letter store(Y/N)  . : N
 Max Transmission Size  . . : 032766     Split Msg(Y/N)  . . . . . : N
 Max TCP/IP Wait  . . . . . : 000000

Channel record displayed.
F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

Figure 12. Channel Definitions screen

From this screen, PF11 activates the Channel Exit Settings screen, which appears as follows:

```
12/17/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
10:45:17                  Channel Exit Settings                     CIC1
MQMMCHN                                                             A000

Channel name . . . : VSE7.TO.OS390X
Channel type . . . : Sender

Send exit name . . : V2OSSNDX
Send exit data . . :

Receive exit name. : V2OSRCVX
Receive exit data. :

Security exit name : V2OSSECX
Security exit data : C0A

Message exit name. :
Message exit data. :




Channel exit settings displayed.
F2=Return  PF3=Quit  PF4=Read  F6=Update
```

Figure 13. Channel Exit Settings screen

The Channel Exit Settings screen allows channel exit programs, and associated data, to be set for send, receive, security and message exits.

Channel exit names can be 1-8 characters, and follow the naming standard for any program defined in the CICS CSD.

Channel exit data can be 1-32 characters and is optional. Data specified in the channel definition is passed to exit programs when they are invoked in the Channel Definition (MQCD) data structure.

Configuration using PCF

Channel definitions can be created and modified using the Programmable Command Formats (PCF).

Channel exits, and their associated data, can be manipulated using the following PCF commands:

- Create Channel
- Change Channel
- Copy Channel
- Inquire Channel

For each of these commands, the following parameters are supported:

SecurityExit (MQCFST)
Security exit name (parameter identifier: MQCACH_SEC_EXIT_NAME). The maximum length of the exit name is restricted to the MQ_EXIT_NAME_LENGTH constant.

MsgExit (MQCFST)
Message exit name (parameter identifier: MQCACH_MSG_EXIT_NAME). The maximum length of the exit name is restricted to the MQ_EXIT_NAME_LENGTH constant.

SendExit (MQCFST)
Send exit name (parameter identifier: MQCACH_SEND_EXIT_NAME). The maximum length of the exit name is restricted to the MQ_EXIT_NAME_LENGTH constant.

ReceiveExit (MQCFST)
Receive exit name (parameter identifier: MQCACH_RCV_EXIT_NAME). The maximum length of the exit name is restricted to the MQ_EXIT_NAME_LENGTH constant.

SecurityUserData (MQCFST)
Security exit user data (parameter identifier: MQCACH_SEC_EXIT_USER_DATA). The maximum length of the exit user data is restricted to the MQ_EXIT_DATA_LENGTH constant.

MsgUserData (MQCFST)
Message exit user data (parameter identifier: MQCACH_MSG_EXIT_USER_DATA). The maximum length of the exit user data is restricted to the MQ_EXIT_DATA_LENGTH constant.

SendUserData (MQCFST)
Send exit user data (parameter identifier: MQCACH_SEND_EXIT_USER_DATA). The maximum length of the exit user data is restricted to the MQ_EXIT_DATA_LENGTH constant.

ReceiveUserData (MQCFST)
Receive exit user data (parameter identifier: MQCACH_RCV_EXIT_USER_DATA). The maximum length of the exit user data is restricted to the MQ_EXIT_DATA_LENGTH constant.

Configuration using MQSC

Channel definitions can be created and modified using MQSeries Commands (MQSC).

Channel exits, and their associated data, can be manipulated using the following MQSC commands:

- DEFINE CHANNEL
- ALTER CHANNEL
- DISPLAY CHANNEL

For each of these commands, the following parameters are supported:

SCYEXIT (string)
Channel security exit name. For MQ/VSE, exit names are 1-8 characters.

MSGEXIT (string)
Channel message exit name. For MQ/VSE, exit names are 1-8 characters.

SENDEXIT (string)
Channel send exit name. For MQ/VSE, exit names are 1-8 characters.

RCVEXIT (string)
Channel receive exit name. For MQ/VSE, exit names are 1-8 characters.

SCYDATA (string)
Channel security exit user data. For MQ/VSE, exit user data can be 0-32 characters.

MSGDATA (string)
Channel message exit user data. For MQ/VSE, exit user data can be 0-32 characters.

SENDDATA (string)
Channel send exit user data. For MQ/VSE, exit user data can be 0-32 characters.

RCVDATA (string)
Channel receive exit user data. For MQ/VSE, exit user data can be 0-32 characters.

Writing and compiling channel-exit programs

Channel exits must be named in the channel definition. You can do this when you first define the channels, or you can add the information later using, for example, the MQSC command ALTER CHANNEL. The format of the exit name must comply with the naming standards for program entries defined in the CICS CSD.

If the channel definition does not contain a user-exit program name, a user exit is not called.

User exits and channel-exit programs are able to make use of all MQI calls, except as noted in the sections that follow. To get the connection handle, an MQCONN must be issued, even though a warning, MQRC_ALREADY_CONNECTED, is returned because the channel itself is connected to the queue manager.

Note: You are recommended to avoid issuing the following MQI calls in channel-exit programs:

- MQCMIT
- MQBACK

An exit runs in the same thread as the MCA itself and uses the same connection handle. So, it runs inside the same UOW as the MCA and any calls made under syncpoint are committed or backed out by the channel at the end of the batch.

Therefore, a channel message exit could send notification messages that will only be committed to that queue when the batch containing the original message is committed. So, it is possible to issue syncpoint MQI calls from a channel message exit.

Channel-exit programs should not modify the Channel data structure (MQCD), except in the case that it is necessary to communicate with Other exit programs via associated user exit data.

Also, for programs written in C, non-reentrant C library function should not be used in a channel-exit program.

All exits are called with a channel exit parameter structure (MQCXP), a channel definition structure (MQCD), a prepared data buffer, data length parameter, and buffer length parameter. The buffer length must not be exceeded:

- For message exits, you should allow for the largest message required to be sent across the channel, plus the length of the MQXQH structure.
- For send and receive exits, the largest buffer you should allow for is 64 KB. Note: Receive exits on sender channels and sender exits on receiver channels use 2 KB buffers for TCP.
- For security exits, the distributed queuing facility allocates a buffer of 1000 bytes.

It is permissible for the exit to return an alternate buffer, together with the relevant parameters. See “MQ_CHANNEL_EXIT - Channel exit” on page 52.

Exit programs in CICS

An exit program must be written in Language Environment (LE) C, COBOL, or PL/I. In CICS, the exits are invoked with EXEC CICS LINK with the Parameters passed by pointers (addresses) in the CICS communication Area (COMMAREA). The exit programs, named in the channel definitions, reside in a library in the LIBDEF SEARCH concatenation of the CICS startup JCL. They must be defined in the CICS system definition file CSD, and must be enabled.

User-exit programs can also make use of CICS API calls, but you should not issue syncpoints because the results could influence units of work declared by the MCA.

Any non-MQSeries for VSE/ESA resources updated by an exit are committed, or backed out, at the next syncpoint issued by the channel program.

Channel-exit calls and data structures

This topic provides reference information about the special MQSeries calls and data structures used when writing channel exit programs. This is product-sensitive programming interface information. You can write MQSeries user exits in LE C, COBOL or PL/I.

In a number of cases, parameters are arrays or character strings whose size is not fixed. For these, a lowercase “n” is used to represent a numeric constant. When the declaration for that parameter is coded, the “n” must be replaced by the numeric

value required. For further information about the conventions used in these descriptions, see the MQSeries Application Programming Reference book.

The calls are:

MQ_CHANNEL_EXIT
Channel exit

The data structures are:

MQCD
Channel data structure

MQCXP
Channel exit parameter structure

MQ_CHANNEL_EXIT - Channel exit

This call definition is provided solely to describe the parameters that are passed to each of the channel exits called by the Message Channel Agent. No entry point called MQ_CHANNEL_EXIT is actually provided by the queue manager; the name MQ_CHANNEL_EXIT is of no special significance since the names of the channel exits are provided in the channel definition MQCD.

MQSeries for VSE/ESA supports four types of channel exit:
- Channel security exit
- Channel message exit
- Channel send exit
- Channel receive exit

The parameters are similar for each type of exit, and the description given here applies to all of them, except where specifically noted.

Syntax:

```
MQ_CHANNEL_EXIT (ChannelExitParms, ChannelDefinition, DataLength,
                 AgentBufferLength, AgentBuffer, ExitBufferLength,
                 ExitBufferAddr)
```

Parameters: The MQ_CHANNEL_EXIT call has the following parameters.

ChannelExitParms (MQCXP) - input/output
Channel exit parameter block. This structure contains additional information relating to the invocation of the exit. The exit sets information in this structure to indicate how the MCA should proceed.

ChannelDefinition (MQCD) - input/output
Channel definition. This structure contains parameters set by the administrator to control the behavior of the channel.

DataLength (MQLONG) - input/output
Length of data. When the exit is invoked, this contains the length of data in the AgentBuffer parameter. The exit must set this to the length of the data in either the AgentBuffer or the ExitBufferAddr (as determined by the ExitResponse2 field in the ChannelExitParms parameter) that is to proceed.

The data depends on the type of exit:
- For a channel security exit, when the exit is invoked this contains the length of any security message in the AgentBuffer field, if ExitReason is MQXR_SEC_MSG. It is zero if there is no message. The exit must set this field to the length of any security message to be sent to its partner if it sets ExitResponse to MQXCC_SEND_SEC_MSG or

MQXCC_SEND_AND_REQUEST_SEC_MSG. The message data is in either AgentBuffer or ExitBufferAddr.

The content of security messages is the sole responsibility of the security exits.

- For a channel message exit, when the exit is invoked this contains the length of the message (including the transmission queue header). The exit must set this field to the length of the message in either AgentBuffer or ExitBufferAddr that is to proceed.
- For a channel send or channel receive exit, when the exit is invoked this contains the length of the transmission. The exit must set this field to the length of the transmission in either AgentBuffer or ExitBufferAddr that is to proceed.

  If a security exit sends a message, and there is no security exit at the other end of the channel, or the other end sets an ExitResponse of MQXCC_OK, the initiating exit is re-invoked with MQXR_SEC_MSG and a null response (DataLength=0).

AgentBufferLength (MQLONG) - input
Length of agent buffer. This can be greater than DataLength on invocation.

For channel message, send, and receive exits, any unused space on invocation can be used by the exit to expand the data in place. If this is done, the DataLength parameter must be set appropriately by the exit.

AgentBuffer (MQBYTE | AgentBufferLength) - input/output
Agent buffer. The contents of this depend upon the exit type:

For a channel security exit, on invocation of the exit it contains a security message if ExitReason is MQXR_SEC_MSG. If the exit wishes to send a security message back, it can either use this buffer or its own buffer (ExitBufferAddr).

For a channel message exit, on invocation of the exit this contains the transmission queue header (MQXQH), which includes the message descriptor (which itself contains the context information for the message), immediately followed by the message data.

If the message is to proceed, the exit can do one of the following:

- Leave the contents of the buffer untouched
- Modify the contents in place (returning the new length of the data in DataLength; this must not be greater than AgentBufferLength)
- Copy the contents to the ExitBufferAddr, making any required changes

Any changes that the exit makes to the transmission queue header are not checked; however, erroneous modifications may mean that the message cannot be put at the destination.

For a channel send or receive exit, on invocation of the exit this contains the transmission data. The exit can do one of the following:

- Leave the contents of the buffer untouched
- Modify the contents in place (returning the new length of the data in DataLength; this must not be greater then AgentBufferLength)
- Copy the contents to the ExitBufferAddr, making any required changes

Note that the first 8 bytes of the data must not be changed by the exit.

ExitBufferLength (MQLONG) - input/output
Length of exit buffer. On the first invocation of the exit, this is set to zero. Thereafter whatever value is passed back by the exit, on each invocation, is presented to the exit next time it is invoked.

ExitBufferAddr (MQPTR) - input/output
Address of exit buffer. This is a pointer to the address of a buffer of storage managed by the exit, where it can choose to return message or transmission data (depending upon the type of exit) to the agent if the agent's buffer is or may not be large enough, or if it is more convenient for the exit to do so.

On the first invocation of the exit, the address passed to the exit is null. Thereafter whatever address is passed back by the exit, on each invocation, is presented to the exit the next time it is invoked.

Usage notes: The function performed by the channel exit is defined by the provider of the exit. The exit, however, must conform to the rules defined here and in the associated control block, the MQCXP.

The ChannelDefinition parameter passed to the channel exit, for MQ/VSE, is always MQCD_VERSION_1.

In general, channel exits are allowed to change the length of message data. This may arise as a result of the exit adding data to the message, or removing data from the message, or compressing or encrypting the message. However, special restrictions apply if the message is a segment that contains only part of a logical message. In particular, there must be no net change in the length of the message as a result of the actions of complementary sending and receiving exits.

For example, it is permissible for a sending exit to shorten the message by compressing it, but the complementary receiving exit must restore the original length of the message by decompressing it, so that there is no net change in the length of the message.

This restriction arises because changing the length of a segment would cause the offsets of later segments in the message to be incorrect, and this would inhibit the queue manager's ability to recognize that the segments formed a complete logical message.

CICS invocation: The MQSeries MCA uses the CICS command level LINK call to pass control to the exit program.

The LINK call passes a communication area (COMMAREA) to the exit program that contains the addresses of the exit parameters as follows:

```
struct tagEXITPARMS
{
    MQCXP    *ChannelExitParms;
    MQCD     *ChannelDefinition;
    MQLONG   *DataLength;
    MQLONG   *AgentBufferLength;
    VOID     *AgentBuffer;
    MQLONG   *ExitBufferLength;
    VOID     *ExitBufferAddr;
} EXITPARMS;
```

MQCD - Channel data structure

The MQCD structure contains the parameters which control execution of a channel. It is passed to each channel exit that is called from a Message Channel Agent (MCA). See "MQ_CHANNEL_EXIT - Channel exit" on page 52.

The MQCD data structure is described in the MQSeries Intercommunication manual.

MQCXP - Channel exit parameter structure

The MQCXP structure is passed to each type of exit called by a Message Channel Agent (MCA). See "MQ_CHANNEL_EXIT - Channel exit" on page 52.

The exit should not expect that any input fields that it changes in the channel exit parameter block will be preserved for its next invocation. Changes made to input/output fields (for example, the ExitUserArea field), are preserved for invocations of that instance of the exit only. Such changes cannot be used to pass data between different exits defined on the same channel, or between the same exit defined on different channels.

The MQCXP data structure is described in the MQSeries Intercommunication manual.

Channel exit sample

MQSeries for VSE/ESA provides a sample channel exit with the installation library. The sample exit is a CICS COBOL program provided in sublibrary member MQPCHNX.Z.

The MQPCHNX sample can be used as a base for your own exit programs. It is generic in the sense that it includes logic to function as a security, send, receive and message exit. It achieves this by examining the contents of the ExitId field in the MQCXP data structure. The ExitId Field indicates which type of exit is being called. Depending on the exit type, the sample branches to appropriate logic.

Alternatively, the sample can be used as a base for individual exit programs that handle only one type of exit call, for example, the message exit. To use the sample in this way, additional logic that examines the ExitId can be removed.

Adopt MCA

The Adopt MCA feature is an integral feature of MQSeries channel operation. It exists to solve a problem with Message Channel Agent (MCA) Receiver tasks falling into an indefinite wait state following a transport error.

When such an error occurs the receiver channel is often unaware of this and remains RUNNING even though the sender is RETRYING.

Once communication is re-established the retrying sender attempts to start a new receiver instance, but since a prior instance of this receiver still exists (because it didn't detect the communication failure), MQSeries "believes" that there has been an invalid attempt to start multiple instances of the same receiver, from the same location, and accordingly treats this as an error, and fails the request.

This problem continues until either the original receiver instance detects the failure, or the channel is forcibly stopped.

Typically an MQSeries receiver is waiting for messages from its sending partner. In the event of a network failure we would hope the receiver (which is effectively in a communication receive call) would be alerted to this by the communication subsystem. In some cases this is not possible and the receiver will continue running indefinitely, even though its partner MCA has ended.

This causes problems when the remote side attempts to re-establish the channel as MQSeries finds the receiver is already running and prevents a duplicate instance from starting up. The channel cannot be restarted until either the operator has manually stopped the orphaned receiver or some communication timeout such as the TCP/IP keepalive timer causes the receiver to eventually fail.

The Adopt MCA feature allows an administrator to specify that MQSeries should automatically stop an orphaned instance of a channel where it receives a new inbound connection request for that channel.

The administrator can specify the level of checking performed before an orphaned candidate is adopted based on combinations of the channel name (must always match for adoption), and the machine address. This allows for less rigorous checking in, for example a DHCP TCP environment where the partner machine's address may change frequently. Note that the Adopt MCA feature is applicable to TCP/IP channels only.

Review section "Features" on page 8 for prerequisites for this feature.

Adopt MCA parameters

Adoption can be enabled or disabled, and the level of checking can be set, via the queue manager's global system definition, MQMT option 1.1, PF9:

```
12/17/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
12:35:25                   Global System Definition                  CIC1
MQMMSYS                     Communications Settings                  A000

    TCP/IP settings                          Batch Interface settings
    TCP/IP listener port : 01461             Batch Int. identifier: MQBISERV
    Licensed clients . . : 00000             Batch Int. auto-start: Y
    Adopt MCA  . . . . . : Y
    Adopt MCA Check  . . : Y

    SSL parameters
    Key-ring sublibrary  : PRD2.SSLKEYS
    Key-ring member  . . : VSE1KEY

    PCF parameters
    System command queue : SYSTEM.ADMIN.COMMAND.QUEUE
    System reply queue . : SYSTEM.ADMIN.REPLY.QUEUE
    Cmd Server auto-start: Y
    Cmd Server convert . : N
    Cmd Server DLQ store : Y

Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 14. Communication Setting, Adopt MCA parameters

The communication setting parameters that affect Adopt MCA feature operation include:

Adopt MCA
Adopt MCA Check

Adopt MCA

The Adopt MCA parameter specifies whether or not an orphaned instance of a channel will be automatically restarted. Valid values include:

YES Automatically stop an orphaned MCA instance, if the appropriate Adopt MCA checks are met.

NO Do not automatically stop an orphaned MCA instance.

The default value is NO.

Activating the Adopt MCA feature by setting this parameter to (Y)es, applies to all TCP/IP Receiver channels.

Adopt MCA Check

The Adopt MCA Check parameter specifies whether the network address of the new MCA must be from the same address as the instance already running. Valid values include:

NO Do not check the new MCA request is from the same network address as the instance that is already running.

YES Check that the new MCA request is from the same network address as the instance that is already running.

The default is NO.

If the Adopt MCA Check parameter is set to (Y)es, the channel will only be adopted if the new MCA request is from the same network address as the instance that is already running.

Bullet-proof channels

MQSeries channels over TCP/IP are difficult to handle when network failures occur. If the TCP/IP connection is broken when an MQSeries channel is active, it is not at all uncommon for the receiving end of the channel to ″hang″ indefinitely in a TCP/IP receive call.

When connectivity is restored, the sender channel is generally unable to reconnect to the hanging receiver. In order to restart the channel, operator intervention is required to forcibly stop the channel. Once this is done, the sending side can normally reconnect.

The circumvention for this problem, on some MQ platforms, has been to use the TCP/IP KeepAlive function by adding a stanza to the qm.ini file (mqs.ini for clients) reading ″TCP: KeepAlive=Yes″. With this stanza in place, MQSeries will enable the SO_KEEPALIVE option on the socket.

This results in TCP/IP itself sending packets across the link from time to time to verify the connection. If enough packets in a row are lost, the connection is presumed to be lost.

From an MQSeries perspective, the receiving side of the channel is notified by TCP/IP that the connection is gone, and thus given a chance to shut down

gracefully. Subsequent reconnection attempts by the sending side of the channel can then proceed normally without operator intervention.

TCP/IP KeepAlive is an excellent solution to this problem, but it has one significant drawback, that is, the KeepAlive timeout interval for connections is generally tunable only on a machine wide basis. In terms of MQSeries channels, a timeout on the order of a few minutes might be reasonable. However, there may be other programs which rely on a timeout of one or two hours. If TCP/IP KeepAlive is the only solution, then MQSeries may not coexist well with these other programs.

Rather than entering a potentially indefinite TCP/IP receive call, and relying on KeepAlive (if it has been properly configured and is in use) to wake up the channel, MQSeries can instead enter a receive call for a finite amount of time. At the end of this time, the queue manager has control to decide whether to receive again or to shut down the channel.

The facility to ″wake up″ channels waiting on a receive call has been named ″bullet-proof channels″.

Although it is the Receiver MCA that is generally waiting for data from the sender, during normal operation, the Sender MCA can be waiting for data from a receiver. In this case, following a communication failure, it is the Sender MCA that can remain in an indefinite wait state. Consequently, the bullet-proof channel feature applies to both sender and receiver channels.

Review section “Features” on page 8 for prerequisites for this feature.

Bullet-proof channel parameters

The bullet-proof channel feature is configurable on a per channel basis. The channel parameter that determines whether a channel will ″wake up″ after a configurable time is the Max TCP/IP Wait parameter.

The Max TCP/IP Wait parameter is configurable from the Channel Record screen, MQMT option 1.3.

```
12/17/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
13:32:11                    Channel Record           DISPLAY         CIC1
MQMMCHN                                                              A000
Channel  : AIX2.TCP.VSE1
 Desc. . :
 Protocol: T (L/T)     Type : R (Sender/Receiver/svrConn)  Enabled : N

Sender
 Remote TCP/IP port . . . . : 00000      LU62 Allocation retry num : 00000000
 Get retry number . . . . . : 00000000   LU62 delay fast (secs). . : 00000000
 Get retry delay (secs) . . : 00000000   LU62 delay slow (secs). . : 00000000
 Convert msgs(Y/N). . . . . : N
 Transmission queue name. . :
 TP name. . :

Sender/Receiver
 Connection :
 Max Messages per Batch . . : 000050     Message Sequence Wrap . . : 999999
 Max Message Size . . . . . : 0004096    Dead letter store(Y/N)  . : N
 Max Transmission Size  . . : 032766     Split Msg(Y/N)  . . . . . : N
 Max TCP/IP Wait  . . . . . : 000300

Channel record displayed.
F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

Figure 15. Channel Record, bullet-proof channel parameter

The Max TCP/IP Wait parameter specifies a period of time (in seconds) for which the channel will wait to receive data from a remote sender. If no data is received within the specified period, the channel will terminate with an error.

By setting the Max TCP/IP Wait parameter to 0, the channel will wait indefinitely to receive data from a remote sender. Effectively, this disables the bullet-proof feature for the channel.

Care must be taken not to specify Max TCP/IP Wait value that is less than the disconnection interval for the channel. The disconnection interval is a parameter of the sender channel definition, and determines how long the sender will keep a channel open when its transmission queue is empty. If the Max TCP/IP Wait value is less than the disconnection interval, the channel will always terminate with an error.

Note: For MQSeries for VSE/ESA, the disconnection interval is equivalent to the Get retry number multiplied by the Get retry delay of the sender channel.

Bullet-proof channel parameters

Chapter 4. System operation

There are four ways of managing an MQSeries for VSE/ESA system:

- You can use the CICS transaction MQMT.

  MQMT allows you to configure, operate, and monitor an MQSeries for VSE/ESA system. MQMT also supports the browsing of message queues and is described in this chapter.
- You can use the MQSeries Command Line interface (MQCL).

  MQCL supports management of queues and channels, and is described in Chapter 5, "Utilities and interfaces," on page 131.
- You can use Programmable Command Format (PCF) messages, as described in Chapter 8, "Programmable system management," on page 167.
- You can use a web browser to access the MQSeries master terminal and associated CICS transactions using the MQSeries for VSE/ESA CICS Web Support (CWS) feature. For more information, refer to "Administration via a web browser" on page 114.

MQSeries master terminal displays

The MQMT menus and display screens are organized in an *informal* hierarchy as depicted in the following diagram. The hierarchy is informal in the sense that non-hierarchical paths between screens can be invoked by using the function keys. For improved legibility, the chart omits certain exit and return paths available from lower level screens.

© Copyright IBM Corp. 1995, 2004

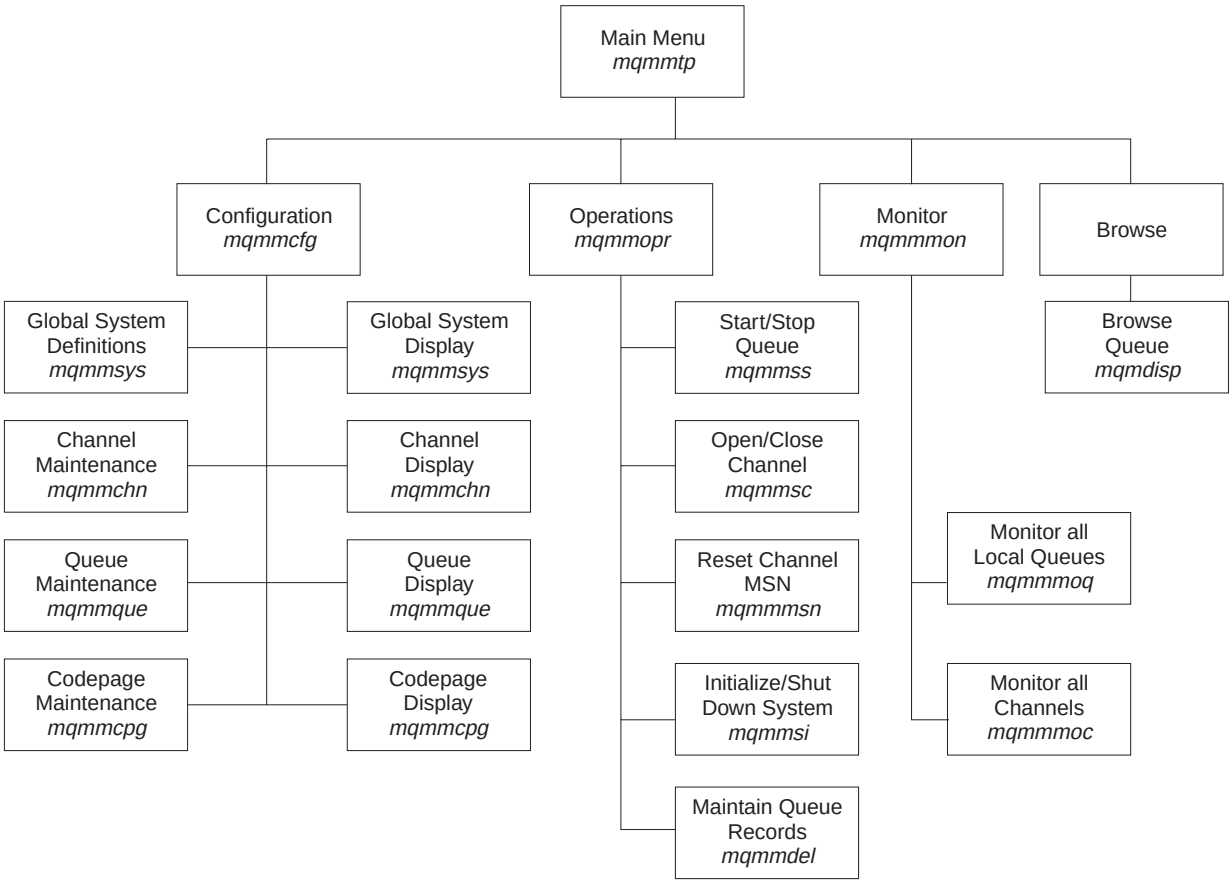

Figure 16. Display screen relationships

The main MQMT menu is shown in "MQSeries master terminal (MQMT) – main menu" on page 63, and the operator functions available through each of the secondary panels are shown in "Configuration functions" on page 65.

General panel layout

MQSeries panels are either menu panels or data entry panels. In either case, they show the following fields:

```
Date         IBM MQSeries for VSE/ESA Version 2.1.2          CICS-Appl
Time                                                         Q-Manager
Panel-Id                                                     Termid









Message line

PF key line 1
PF key line 2 (if necessary)
```

Figure 17. General panel layout

Where:

CICS-Appl
: The VTAM application ID for this CICS partition.

Panel-Id
: The name of the displayed panel.

Q-Manager
: The name of the MQSeries queue manager specified in the global definitions.

Termid
: The ID of the CICS terminal on which this panel is displayed.

MQSeries master terminal (MQMT) – main menu

You can invoke the MQSeries system administrator program, MQMT, from any 3270 terminal. To access the operator functions, type `MQMT` at the CICS prompt.

When MQMT starts, the main menu is displayed.

```
07/06/2002        IBM MQSeries for VSE/ESA Version 2.1.2            MQBDTS
10:55:25           *** Master Terminal Main Menu ***                CIC1
MQMMTP                                                              A001
                          SYSTEM IS ACTIVE

                            1. Configuration

                            2. Operations

                            3. Monitoring

                            4. Browse Queue Records


                         Option:









Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998,2002. All Rights Reserved.
    CLEAR/PF3 = Exit                           ENTER=Select
```

Figure 18. Master terminal main menu

From the main menu, one of several submenus can be selected. The first three selections correspond to broad categories that include most MQSeries operator functions:

- Configuring MQSeries
- Operating (controlling) MQSeries
- Monitoring MQSeries

The fourth function allows you to display the records on a selected queue:

- Browsing MQSeries queues

Each submenu presents a list of operator functions available from that screen. When a specific function is selected, the appropriate data entry or data display screens are presented to the operator.

Master Terminal transactions

The functions of the MQSeries system administrator program can be invoked directly using the following transaction code table. For those customers using an External Security Manager, specific functions can be restricted to certain users or class of users. Alternatively, administration tasks can be restricted by enabling command and command resource security (for more information refer to “Command security” on page 287 and ’Command resource security’ “Command security” on page 287.

```
MQMT  Master Terminal Main Menu
   |===> MQMC  Configuration Main Menu
   |        |===> MQMS  Global System Definition \
   |        |===> MQMQ  Queue Main Options        \maintenance
   |        |===> MQMH  Channel Record             \mode
   |        |===> MQMP  Code Page Definition        \
   |        |===> MQDS  Global System Definition \
   |        |===> MQDQ  Queue Main Options        \display
   |        |===> MQDH  Channel Record             \mode
   |       +===> MQDP  Code Page Definition        \
   |
```

```
|===> MQMM  Monitor Main Menu
|       |===> MQQM  Monitor Queues
|       +===> MQCM  Monitor Channels
|
|===> MQMO  Operations Main Menu
|       |===> MQMA  Start / Stop Queue
|       |===> MQMB  Open / Close Channel
|       |===> MQMR  Reset Channel Message Sequence
|       |===> MQMI  Initialization / Shutdown of System
|       +===> MQMD  Maintain Queue Message Records
|
+===> MQBQ  Browse Queue Records
```

Operator screen action keys

The action keys available on each MQSeries operator screen are displayed at the bottom of the screen with an explanation of their function. In general, the following keys are available and associated with the indicated action:

CLEAR – Exit MQMT
PF2 – Return to previous menu
PF3 – Exit to CICS
PF4 – Select/Read (Same as Return or Enter keys)
PF5 – Add
PF6 – Update
PF7 – Backward
PF8 – Forward
PF9 – Screen-dependent
PF10 – Screen-dependant
PF11 – Screen-dependant
PF12 – Delete

Configuration functions

Selecting option 1 (Configuration) from the master terminal main menu (see Figure 18 on page 64) displays the following screen:

```
12/24/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
09:45:52           *** Configuration Main Menu ***                  CIC1
MQWMCFG                                                             A001
                           SYSTEM IS ACTIVE

                         Maintenance Options :
                             1. Global System Definition
                             2. Queue Definitions
                             3. Channel Definitions
                             4. Code Page Definitions

                         Display Options     :
                             5. Global System Definition
                             6. Queue Definitions
                             7. Channel Definitions
                             8. Code Page Definitions

                      Option:



Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Process  PF2=Return  PF3=Exit
```

Figure 19. Configuration main menu

On this screen, selections 1, 2, 3, and 4 allow you to perform maintenance functions on various MQSeries configuration objects. Selections 5, 6, 7 and 8 allow viewing of the same objects.

Global system definition

Before you can do anything with messages and queues, you must configure a queue manager. Once the installation process is complete, you use the MQSeries "Global System Definition" screen to configure the queue manager and start it.

Default objects form the basis of any object definitions that you make. System objects are required for queue manager operation and you must create these objects for the queue manager that you created.

Guidelines for configuring queue managers

A queue manager manages the resources associated with it, in particular the queues that it owns. It provides queuing services to applications for Message Queuing Interface (MQI) calls and commands to create, modify, display, and delete MQSeries objects. Some tasks you must consider when creating a queue manager are:

- Selecting a unique queue manager name, as described on Page 66.
- Creating the dead-letter and system log queues, as described on Page 67.
- Backing up the configuration file, as described on Page 75.

The tasks in this list are explained in the sections that follow.

Specifying a unique queue manager name

When you create a queue manager, you must ensure that no other queue manager has the same name, anywhere in your network. Queue manager names are not checked at create time, and non-unique names will prevent you from using channels for distributed queuing.

One method of ensuring uniqueness is to prefix each queue manager name with its own (unique) node name. For example, if a node is called `accounts`, you could name your queue manager `accounts.saturn.queue.manager`, where `saturn` identifies a particular queue manager and `queue.manager` is an extension you can give to all queue managers. Alternatively, you can omit this, but note that `accounts.saturn` and `accounts.saturn.queue.manager` are *different* queue manager names.

If you are using MQSeries for communicating with other enterprises, you can also include your own enterprise as a prefix. We do not actually do this in the examples, because it makes them more difficult to follow.

Specifying the dead-letter and system log queues

The dead-letter queue is a local queue where messages are put if they cannot be routed to their correct destination.

Attention: It is vitally important to have a dead-letter queue on each queue manager in your network. Failure to do so may mean that errors in application programs cause channels to be closed or that replies to administration commands are not received.

You create a dead-letter queue as a local queue; “Creating local queues” on page 77 for details.

For example, if an application attempts to put a message on a queue on another queue manager, but the wrong queue name is given, the channel is stopped, and the message remains on the transmission queue. Other applications cannot then use this channel for their messages.

The channels are not affected if the queue managers have dead-letter queues. The undelivered message is simply put on the dead-letter queue at the receiving end, leaving the channel and its transmission queue available.

Therefore, when you create a queue manager you should specify the name of the dead-letter queue.

Similarly, the system log queue is essential for normal queue manager operation. The system log is used by the queue manager to report diagnostic and error messages. Some informational messages are generated when the queue manager is started, consequently, the system log queue should be defined to the queue manager before the system is started for the first time.

Like the system dead-letter queue, the system log is an MQSeries queue and should be defined as a local queue.

Configuring the queue manager

For each installation of the MQSeries system, one (and only one) queue manager must be defined. This is accomplished through the screen shown in Figure 20 on page 68. This screen is also used to modify the default or previously defined global parameters.

```
12/24/2002          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQ212
10:16:45                  Global System Definition                   CIC1
MQWMSYS                 Queue Manager Information                    A001
Queue Manager . . . . . .: VSE.TS.QM1
Description Line 1. . . . .:
Description Line 2. . . . .:
                        Queue System Values
Maximum Number of Tasks . .: 00001000       System Wait Interval : 00000030
Maximum Concurrent Queues .: 00001000       Max. Recovery Tasks  : 0000
Allow TDQ Write on Errors  : Y    CSMT      Allow Internal Dump  : Y
                        Queue Maximum Values
Maximum Q Depth . . . . . .: 00640000       Maximum Global Locks.: 00001000
Maximum Message Size. . . .: 00004096       Maximum Local Locks .: 00001000
Maximum Single Q Access . .: 00000100
                        Global QUEUE /File Names
Local Code Page . . : 01047
Configuration File. : MQFCNFG
LOG Queue Name. . . : SYSTEM.LOG
Dead Letter Name. . : SYSTEM.DEAD.LETTER.QUEUE
Monitor Queue Name. : SYSTEM.MONITOR

Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF6=Upd  PF9=Comms  PF10=Log   PF11=Event
```

Figure 20. System queue manager information

On this screen, the data entry fields are:

Queue Manager
This is the name of the local queue manager for this MQSeries system installation. The name may be up to 48 characters and must conform to the MQSeries naming requirements.

Description Lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Queue System Values

Maximum Number of Tasks
The maximum number of simultaneous connections to the queue manager.

Maximum Concurrent Queues
The maximum number of simultaneously open queues.

Allow TDQ Write on Errors
Y – allow writes to the CICS TDQ 'CSMT' if SYSTEM.LOG not available

N – do not allow write to the CICS TDQ

B – write to both SYSTEM.LOG and the CMST TDQ.

System Wait Interval
The sleep time in seconds for the system monitor program and startup of trigger programs after system initialization.

Max. Recovery Tasks
Maximum number of tasks attached by the system monitor when errors are detected in queues or control blocks attached to queues. A high number would lead to the use of too many CICS resources and have a negative impact on the overall CICS performance. The suggested value is zero.

Allow Internal Dump
Allow the MQSeries API to process a CICS Task Dump if the internal areas are corrupted.

Queue Maximum Values

Maximum Q Depth
The maximum number of records that will be left unread on a queue.

Maximum Message Size
The maximum size of any message.
Maximum Single Q Access
The maximum number of object handles allowed for a queue.
Maximum Global Locks
The maximum number of entries that the queue manager uses to maintain destructive PUT or GET locks, per queue, for the system.
Maximum Local Locks
The maximum number of entries that an application can use to maintain destructive PUT, or GET locks, per queue, for each individual task.

Global QUEUE /File Names

Local Code Page
The code page in use on the local system. If you plan to support remote client connections, you must use a local code page that can be translated into the code page of the remote client system. Generally, code page 1047 is a good choice, because many translations for this code page are provided with LE. Alternatively, you can define your own translation tables (see Appendix F, "MQSeries clients," on page 465) and set the local code page appropriately.
Configuration File
The CICS file definition name of the MQSeries configuration file.
LOG Queue Name
The queue name where the MQSeries programs write information and error messages. This is the system log queue.
Dead Letter Name
The file where channel programs write messages that are received with the wrong queue manager name or queue name. These messages will have the dead letter header placed in front of the queue record.
Monitor Queue Name
Diagnostic queue for MQI monitoring. The MQI monitor can be activated using MQMT option 2.1 (for more details refer to "Queuing System Request" on page 103.)

Note: Queue maximum value fields restrict the allowed values in the queue definition field values, while the rest of the fields affect the run-time values when the System is initialized.

Queue Manager Communications Settings

Press PF9 (Comms) on the Global System Definition screen to display the Queue Manager Communications Settings screen:

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
14:47:42                  Global System Definition                  CIC1
MQMMSYS                    Communications Settings                  A000

    TCP/IP settings                          Batch Interface settings
    TCP/IP listener port : 01414             Batch Int. identifier: MQBISERV
    Licensed clients . . : 00000             Batch Int. auto-start: Y
    Adopt MCA  . . . . . : Y
    Adopt MCA Check  . . : Y

    SSL parameters
    Key-ring sublibrary  : PRD2.SSLKEYS
    Key-ring member  . . : MQVSEKEY

    PCF parameters
    System command queue : SYSTEM.ADMIN.COMMAND.QUEUE
    System reply queue . : SYSTEM.ADMIN.REPLY.QUEUE
    Cmd Server auto-start: Y
    Cmd Server convert . : N
    Cmd Server DLQ store : Y

Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 21. Queue manager communications settings

TCP/IP settings:

TCP/IP listener port
The port number that MQSeries uses for accepting TCP/IP connection requests from remote queue managers and MQ clients. The default port value is 1414, however any unreserved port number can be used.

Licensed clients
The number of clients for which the MQSeries system is registered. This represents the maximum number of concurrent remote client connections that the local system will support at any one time. Use the number from your MQSeries for VSE/ESA license documentation.

Adopt MCA
Indicates whether an Message Channel Agent (MCA) should adopt another MCA process if one is already running. For more information about the Adopt MCA feature, see "Adopt MCA" on page 55.

Adopt MCA Check
Indicates whether the network address of an existing MCA should be checked before adopting the MCA process. For more information, see "Adopt MCA" on page 55.

SSL parameters:

Key-ring sublibrary
The key-ring sublibrary identifies the VSE sublibrary name that contains the private key and certificate intended for use by SSL enabled MQ channels. This is the SSL product KEYLIB or key ring file name. Queue managers that require SSL enabled channels, must identify a valid key-ring sublibrary.

Key-ring member
The key-ring member is the key-ring sublibrary member name of the SSL

private key (.PRVK) and certificate (.CERT) files. Queue managers that require SSL enabled channels, must identify a valid key-ring member name.

For more information regarding SSL features, refer to Chapter 10, "Secure Sockets Layer services," on page 279.

PCF parameters:

System command queue
The command queue where local and remote administration applications can place PCF messages to be processed by the local queue manager. The default value for this parameter is: SYSTEM.ADMIN.COMMAND.QUEUE.

For more information, refer to Chapter 8, "Programmable system management," on page 167.

System reply queue
The command reply queue used by the MQSC batch utility program. For more information, refer to Chapter 9, "MQSeries commands," on page 253.

Cmd Server auto-start
Indicator for the automatic activation of the PCF command server. This parameter can have the following values:

Y Automatically start the PCF command server when the queue manager is initialized.

N Do not automatically start the PCF command server when the queue manager is initialized.

Cmd Server convert
Indicator for the data conversion of PCF messages by the command server. This parameter can have the following values:

Y Apply data conversion to PCF messages retrieved by the command server.

N Do not apply data conversion to PCF messages retrieved by the command server.

Cmd Server DLQ store
Indicator for the storage of undeliverable PCF reply messages to the system dead letter queue by the command server. This parameter can have the following values:

Y Command server will attempt to place undeliverable PCF response messages to the system dead letter queue.

N Command server will not attempt to place undeliverable PCF response messages to the system dead letter queue.

Batch Interface settings:

Batch Int. identifier
Batch interface identifier. This is a 1-8 character identifier used by batch MQ applications to connect to a relevant queue manager. The identifier must be unique within the context of a VSE/ESA system. The default value for this parameter is MQBISERV.

For more information refer to "Using the batch interface" on page 136.

Batch Int. auto-start
Indicator for the automatic activation of the batch interface. This parameter can have the following values:

Y Automatically start the batch interface when the queue manager is initialized.

N Do not automatically start the batch interface when the queue manager is initialized.

Queue Manager Log and Trace Settings

Press PF10 (Log) on the Global System Definition screen to display the Queue Manager Log and Trace Settings screen:

```
12/24/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
11:11:23                  Global System Definition                  CIC1
MQWMSYS                    Log and Trace Settings                   A001

           Log Settings     Q C              Trace Settings

      Informational . . . : Y N          MQI calls . . . . . : N
      Warning . . . . . : Y N            Communication . . . . : N
      Error . . . . . : Y Y              Reorganization  . . . : N
      Critical  . . . . . : Y R          Data conversion . . . : N
                                         System  . . . . . : N
           - and/or -

      Communication . . . : Y N
      Reorganization  . . : Y N
      System  . . . . . : Y N




Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 22. Queue manager log and trace settings

Logging, in this sense, refers to the type or severity of messages written to the SYSTEM.LOG. Tracing refers to entries written to the CICS auxiliary trace. Configurability is intended to reduce certain processing overhead involved with logging and tracing under MQ/VSE. It is expected that many customers, in a production environment, will reduce logging to error and critical messages only, and switch tracing off altogether. You can also view log and trace settings from MQMT option 1.5.

Log Settings: Log settings involve a choice between logging by the severity of messages and/or the general type of message. For example, it is possible to select logging for error and critical messages only, along with, for example, general system messages. This is possible with the following log settings:

Informational	N
Warning	N
Error	Y
Critical	Y
and/or	
Communication	N
Reorganization	N

System Y

With this configuration, all general system messages would be written to the SYSTEM.LOG (including informational and warning messages), otherwise, only error and warning messages would be written to the log.

Log settings are made under the column labelled ″Q″ (for Queue). Valid values include:

N Suppress messages of this severity/type.

Y Send messages of this severity/type to the system log queue.

Diagnostic and error messages can optionally be sent to the VSE/ESA console. A message cannot be sent to the console unless it is also sent to the system log queue. Consequently, for example, it is not possible to suppress informational messages and also have them sent to the VSE/ESA console.

Settings for optional logging to console are made under the column labelled ″C″ (for Console). Valid values include:

N Do not send messages of this severity to the console.

Y Send messages of this severity to the console.

R Send messages of this severity to the console and prompt for an operator reply.

Messages sent to the console are prefixed with the generic MQSeries message identifier MQI0200I, followed by the message identifier and text of the message written to the system log queue. The MQI0200I message is truncated to a single console line if necessary.

Messages sent to the console requiring an operator reply are high-lighted and remain on the VSE/ESA console until an operator reply is registered. The text “...awaiting reply” as appended to messages sent to the console that require and operator response.

Care should be taken not to flood the VSE/ESA console with messages (particularly messages requiring an operator response). To avoid flooding the console, it is recommended that a setting of ″R″ (for Reply) is only used for Critical messages.

Trace Settings: Trace settings involve selection by general type. For example, it is possible to trace communications programs and general system programs, and exclude tracing for MQI calls, reorganization and data conversion. This example is possible with the following trace settings:

MQI calls N
Communication Y
Reorganization N
Data conversion N
System Y

Normally, tracing is only required when a serious system problem has been encountered, and IBM service personnel have requested a trace of MQ system activity. Since tracing involves some system overhead, it is recommended that during normal operation, tracing is deactivated (that is, set all selections to “N”).

Queue Manager Event Settings

Press PF11 (Event) on the Global System Definition screen to display the Queue Manager Event Settings screen:

```
11/11/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
13:20:58                   Global System Definition                  CIC1
MQMMSYS                         Event Settings                       A005

    Event queues
    Queue manager events : SYSTEM.ADMIN.QMGR.EVENT
    Channel events . . . : SYSTEM.ADMIN.CHANNEL.EVENT
    Performance events . : SYSTEM.ADMIN.PERFM.EVENT

    Qmgr events         Channel events      Performance events
    Inhibit  . . . : Y  Started  . . . : Y  Queue depth  . . : N
    Local  . . . . : Y  Stopped  . . . : Y  Service interval : N
    Remote . . . . : Y  Conversion err : Y
    Authority  . . : Y
    Start/Stop . . : Y









Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 23. Queue Manager event settings

Event queues:

Queue manager events
The queue manager events queue names the target queue for queue manager event messages. The name specified should be a valid local queue. The MQJQUEUE.Z files contains sample VSAM definitions for files to host the event queues. The default name for the queue manager event queue is SYSTEM.ADMIN.QMGR.EVENT, and the default VSAM file for this queue is MQFIEQE.

Channel events
The channel events queue names the target queue for channel event messages. The name specified should be a valid local queue. The MQJQUEUE.Z files contains sample VSAM definitions for files to host the event queues. The default name for the channel event queue is SYSTEM.ADMIN.CHANNEL.EVENT, and the default VSAM file for this queue is MQFIECE.

Performance events
The performance events queue names the target queue for performance event messages. The name specified should be a valid local queue. The MQJQUEUE.Z files contains sample VSAM definitions for files to host the event queues. The default name for the performance event queue is SYSTEM.ADMIN.PERFM.EVENT, and the default VSAM file for this queue is MQFIEPE.

Qmgr events:

Inhibit
Inhibit events indicate that an MQPUT or MQGET operation has been attempted against a queue, where the queue is inhibited for puts or gets respectively.

Local
Local events indicate that an application (or the queue manager) has not been able to access a local queue, or other local object. For example, when an application attempts to access an object that has not been defined.

Remote
Remote events indicate that an application (or the queue manager) cannot access a (remote) queue on another queue manager. For example, when the transmission queue to be used is not correctly defined.

Authority
Authority events indicate that an authorization violation has been detected. For example, an application attempts to open a queue for which it does not have the required authority, or a command is issued from a user ID that does not have the required authority.

Start/Stop
Start and stop events indicate that a queue manager has been started or has been requested to stop

Channel events:

Started
Channel started events are generated when a Sender or Receiver channel starts. Channel started events are not generated for Client (SVRCONN) channels.

Stopped
Channel stopped events are generated when a Sender or Receiver channel stops. Channel stopped events are not generated for Client (SVRCONN) channels.

Conversion err
Channel conversion error events are generated by Sender channels when an MQGET call to the transmission queue fails with a data conversion error.

Performance events:

Queue depth
Queue depth events are related to the queue depth, that is, the number of messages on the queue. The types of queue depth events are: Queue Depth High, Queue Depth Low and Queue Full.

Service interval
Queue service interval events indicate whether a queue was “serviced” within a user-defined time interval called the service interval. Depending on the circumstances, queue service interval events can be used to monitor whether messages are being taken off queues quickly enough.

Backing up the configuration file after creating the queue manager

When you create the queue manager, the queue manager configuration file MQFCNFG is updated. This contains configuration parameters for the queue manager, and the queue and channel definitions.

You should make a backup of this file. If you have to create another queue manager, perhaps to replace the existing queue manager if it is causing problems, you can reinstate the backup when you have removed the source of the problem.

If you restore the configuration file then be aware that you will reset all channel MSNs to the values they held at the time of the backup. You may need to reset the channel MSNs using the master terminal transaction or by using batch utility MQPUTIL.

Queue definitions

Selecting 2 on the configuration menu allows you to maintain (add, modify, or delete) queue definitions for the local installation of MQSeries.

Note: The same screens are used to accomplish the three functions of adding, modifying, or deleting a queue definition; the required action being selected through the function keys. The following sections present the screens that you see if you are adding a new queue definition.

"Modifying and deleting queue definitions" on page 86 explains how you modify or delete a queue.

To create a queue definition, multiple screens may be involved. The first screen is the same for all queues, and allows entry of the queue name and type.

Based on the type you enter, an appropriate second screen is displayed for you to enter the remainder of the data to complete the definition.

In the case of local queues, a third screen, the Extended Queue Definition Screen can be displayed.

The first screen displayed is as shown in Figure 24.

```
12/24/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:34:45                Queue Main Options                          CIC1
MQWMQUE                                                             A001
                             SYSTEM IS ACTIVE

       Default Q Manager. : VSE.TS.QM1

       Object Type. . . . :        L = Local Queue
                                   R = Remote Queue
                                   AQ = Alias Queue
                                   AM = Alias Queue Manager
                                   AR = Alias Reply Queue

       Object Name. . . . :









PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF12=Delete
```

Figure 24. Queue main options screen

On this screen, the data entry fields are:

Object Type
This is a two character field with the acceptable entries listed on the screen. The type determines the next screen to be displayed.

Object Name
This is the name of the queue (or alias) being defined. The name may be up to 48 characters, must be unique among all other defined queues for this installation, and must conform to the MQSeries naming requirements.

The Object Type you select on this screen is used to determine which of the definition screens is displayed:

- L selects a local queue definition; see "Creating local queues"
- R selects a remote queue definition; see "Create remote queue" on page 83
- AQ selects an alias queue definition; see "Create alias queue" on page 84
- AM selects an alias manager definition; see "Create alias queue manager" on page 85
- AR selects an alias reply queue definition; see "Create alias reply queue" on page 86

Creating local queues

Selecting "L" on the screen in Figure 24 on page 76 displays the screen shown in Figure 25.

```
12/24/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:40:01               Queue Definition  Record                     CIC1
MQWMQUE           QM - VSE.TS.QM1                                   A001

                      Local Queue Definition

Object Name. . . . . . . . : ANYQ
Description line 1 . . . . : Test queue
Description line 2 . . . . :

Put Enabled  . . . . . . . : Y   Y=Yes, N=No
Get Enabled  . . . . . . . : Y   Y=Yes, N=No

Default Inbound status . . : A   A=Active,I=Inactive
        Outbound status. . : A   A=Active,I=Inactive

Dual Update Queue. . . . . :

Automatic Reorganize (Y/N) : N   Start Time. : 0000    Interval. . :  0000
VSAM Catalog . . . . . . . :

Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF10=Queue  PF12=Delete
```

Figure 25. Local queue definition

On this screen, the data entry fields are:

Object Name
Filled in from the previous screen.

Description lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Put Enabled
A toggle that enables or disables MQPUT operations against this queue.

Get Enabled
A toggle that enables or disables MQGET operations against this queue.

Default Inbound status
Sets the initial status to active (A) or inactive (I) at run time for the inbound direction of the queue.

Default Outbound status
Sets the initial status at run time for the outbound direction of the queue.

Dual Update Queue
When an existing queue name is entered here, dual queuing is activated. The queue being created becomes the primary queue, and the queue entered in this field becomes the dual queue. The definition of the dual queue is updated automatically with the name of the primary queue. The queue display of the dual queue has a corresponding heading "Dual Source Queue".

Dual Source Queue
The name of the primary queue, for which the queue being displayed is the dual queue. This field appears only when a local queue serves as a dual update queue.

Note: When an existing queue is defined as the dual to a primary queue, these two queues both participate in the same logical units of work.

If, for any reason, it becomes impossible to update the dual queue (for example, if the queue becomes disabled, the associated file is closed, or an ISC link is lost), updates continue to be made to the primary queue and the dual queue goes to a recovery status.

Automatic Reorganize
A toggle that enables or disables automatic reorganization of the VSAM file associated with the selected queue, at specified time intervals.

Notes:

1. MQSeries for VSE/ESA uses VSAM files to move messages. The indexes of these files can become fragmented, causing the performance of the system to suffer.

   To reorganize these files you must use VSAM utilities, and the Automatic Reorganize feature automates the process.

   If automatic reorganization is not suitable then use the batch reorganization utility MQPREORG.
2. You are recommended to use the Automatic Reorganize feature only for queue files with high activity.
3. When you specify a queue file that is to be automatically reorganized, you should ensure that there is only one MQSeries queue associated with each VSAM file.
4. The Start Time identifies the time of day when the reorganization runs. For example, 0230 is 2:30 a.m., and 2345 is 11:45 p.m..
5. The Interval, in minutes, identifies the frequency that the reorganization runs. For example, 1440 is daily, and 0120 is every two hours.
6. The VSAM Catalog identifies the VSAM Catalog of the VSAM file that contains the queue.

7. The reorganization can take place even when there are messages on the queue. Message data is retained. However, logically deleted messages are removed.
8. Be aware that during reorganization, application programs do not have access to the queue. If there are any active applications, the reorganization will retry or postpone. Applications that attempt to access the queue after the reorganization starts will wait. The processing time for the reorganization varies, depending on the number of unprocessed messages remaining on the queue. We recommend that you attempt to clear queues before the reorganization time to minimize the unavailability of the queue.
9. If specifying automatic reorganization for a queue then it is required that the VSAM file containing the queue be defined with cluster data name with suffix of .DATA and index name with suffix of .INDEX. This is also required for the MQFREOR file.

Local queue extended definition screen

By pressing function key 10 (PF10), you can display a second screen to enter the extended definition fields for the queue. In the case of a request to add a queue, this Extended Definition Screen is presented automatically. This detailed screen is shown in Figure 26:

```
11/11/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
 13:00:08               Queue Extended Definition                     CIC1
 MQMMQUE                                                              A005

 Object Name: ANYQ

 General               Maximums                    Events
 Type   . . : Local    Max. Q depth . : 00001000  Service int. event: N
 File name  : MQFI001  Max. msg length: 00004000  Service interval  : 00000000
 Usage  . . : N        Max. Q users . : 00000100  Max. depth event  : N
 Shareable  : Y        Max. gbl locks : 00000200  High depth event  : N
                       Max. lcl locks : 00000200  High depth limit  : 000
                                                  Low depth event . : N
 Triggering                                       Low depth limit . : 000
 Enabled  . : N        Transaction id.:
 Type . . . :          Program id . . :
 Max. starts: 0001     Terminal id  . :
 Restart  . : N        Channel name . :
 User data  :
            :

 Requested record displayed.
 PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
             PF9=List  PF10=Queue
```

Figure 26. Local queue extended definition

On this screen, the data entry fields are:

Object Name
Filled in from the previous screen. Cannot be modified.

Physical Queue Information

Usage Normal (N) means that the queue is used by an application to receive inbound messages. Transmission (T) means that the queue is used by MQSeries to hold outbound messages destined for another MQSeries queue manager.

Shareable
Defines a queue as shareable or exclusive on input.

File name
The CICS file name, with a maximum of seven characters, used to store messages for this queue.

Note: Do not use the MQFREOR file for queue definitions. This file is used by the automatic VSAM reorganization feature, and is deleted and redefined during reorganization. Consequently, message data for queues defined in MQFREOR will be lost.

Maximum Values

Max. Q depth
The maximum number of messages allowed on this queue. The default value is the value specified in the global system definition.

Max. msg length
The maximum length of an application message processed on this queue.

Max. Q users
The maximum number of MQOPEN calls that can occur for this queue simultaneously.

Max. gbl locks
Allocates the locking table for this queue for all committed MQGET calls.

Max. lcl locks
This is used to allocate the locking table for this queue for each task's noncommitted MQGET calls.

Trigger Information

Enabled
If you are defining a transmission queue for use with a sender channel, set this value to Yes (Y). Otherwise, for use with a server or receiver channel, set this field to No (N).

Alternatively, for queues with a usage of (N)ormal, set this parameter to (Y)es to instruct the queue manager to automatically start a trigger process instance in response to message activity on the queue.

If no trigger process is required, set this parameter to (N)o.

Type

F A trigger is generated when the queue goes from an empty to a non-empty state.

E A trigger instance is generated every time a message is placed on the queue. This can be constrained by the "Maximum Trigger Starts" parameter which sets a maximum number of trigger instances associated with the queue that can be running at any one time.

Only one transaction can be active against the queue if the Trigger Type is set to F.

Max. starts
The maximum number of trigger threads that can be active at once.

Restart
Allows the automatic restart of an application if the trigger count goes to zero. It restarts one trigger if messages are available on this queue.

Transaction id
The name of the transaction to be started by a trigger, with a length of four characters. If a transaction ID is specified, this transaction will be started. For a transmission queue, this field is left blank.

If a transaction identifier is defined, the program identifier should be left blank.

Once the initial maximum trigger starts is reached then MQSeries for VSE/ESA only checks that the maximum trigger starts are running at every system interval and not when each task completes. If it is important to have a definite number of trigger instances running against a queue, use Program ID to identify your trigger program.

Program id
The name of the user program to be invoked, with a length of eight characters. If you are defining a transmission queue to be used with a sender channel, MQPSEND must be used. If the field for Trans ID is left blank and this field contains a program ID, the specified program is linked.

Terminal id
Optional field of four characters used for problem determination. It is attached to the transaction ID specified in the Trans ID field.

Channel Name
Is the channel name, with a maximum of 20 characters.

This parameter is relevant for transmission queues only, and is used by the MQSeries Sender MCA (MQPSEND) to identify the appropriate channel for transmitting messages to a remote MQSeries system.

User Data
A field for static data to be passed to the trigger instance. When a trigger instance is activated, it is passed data in the form of the MQTM structure (see the CMQTML and CMQTMV copybooks). Data in the User Data field is passed in the MQTM-USERDATA field of the MQTM structure. The trigger instance can use this data for its own internal logic.

Event Information

Service int. event
Specifies whether service interval events are required for this particular queue. Queue service interval events indicate whether a queue was 'serviced' within a user-defined time interval called the service interval. Depending on the circumstances at your installation, you can use queue service interval events to monitor whether messages are being taken off queues quickly enough.

There are two types of queue service interval event:

A Queue Service Interval OK event, which indicates that, following an MQPUT call or an MQGET call that leaves a non-empty queue, an MQPUT call or an MQGET call was performed within a user-defined time period, known as the service interval.

A Queue Service Interval High event, which indicates that, following an MQGET or MQPUT call that leaves a non-empty queue, an MQGET call was not performed within the user-defined service interval. This event message can be caused by an MQPUT call or an MQGET call.

To enable both Queue Service Interval OK and Queue Service Interval High events you need to set the QServiceIntervalEvent control attribute to High. Queue Service Interval OK events are automatically enabled when a Queue Service Interval High event is generated. You do not need to enable Queue Service Interval OK events independently.

These events are mutually exclusive, which means that if one is enabled the other is disabled. However, both events can be simultaneously disabled.

Valid values for this parameter include:
H - High
O - Ok
N - None

Service interval
Specifies the interval used to determine whether the performance events High and Ok are generated. The value specified represents milliseconds.

Max. depth event
Indicates whether or not queue full events are required. A queue full event is generated when an attempt to put a message to the queue fails because the queue has reached its maximum queue depth.

High depth event
Indicates whether or not Queue Depth High events are required. A Queue Depth High event is generated when a message is put to the queue which increases the queue depth to the limit specified by the High depth limit parameter.

High depth limit
Specifies a queue depth as a percentage that is to be used to determine whether or not a Queue Depth High event should be generated. For example, a value of 80 for a queue with a maximum queue depth of 1000, would mean a Queue Depth High is generated when the queue depth reaches 800 messages.

Low depth event
Indicates whether or not Queue Depth Low events are required. A Queue Depth Low event is generated when a message is retrieved from the queue which decreases the queue depth to the limit specified by the Low depth limit parameter.

Low depth limit
Specifies a queue depth as a percentage that is to be used to determine whether or not a Queue Depth Low event should be generated. For example, a value of 20 for a queue with a maximum queue depth of 1000, would mean a Queue Depth Low is generated when the queue depth reduces to 200 messages.

Notes:

1. The PF10 key can be used to toggle between the Local Queue Definition screen (Figure 25 on page 77) and the Local Queue Extended Definition screen (Figure 26 on page 79).
2. Either Trans ID or Program ID is required if triggering is enabled.
3. The internal MQSeries trigger API transaction MQ02 cannot be used as a trigger transaction ID.

4. Both a trigger transaction and a trigger program can be defined, but only the trigger transaction is activated and the trigger program name is passed in the trigger communications area.
5. The queue maximums are restricted by the queue manager's global maximums, and the maximum limits of the product.

Navigation through the screens is dependent upon the PF keys.

Create remote queue

Selecting "R" on the screen in Figure 24 on page 76 displays the screen shown in Figure 27.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
10:00:38               Queue Definition  Record                     CIC1
MQWMQUE           QM - VSE.TS.QM1                                   A002

                      Remote Queue Definition

Object Name. . . . . . . . : VSE1.NT1.RQ1
Description line 1 . . . . : Remote queue to LQ1 on NT1
Description line 2 . . . . :

Put Enabled  . . . . . . . : Y   Y=Yes, N=No
Get Enabled  . . . . . . . : Y   Y=Yes, N=No

Remote Queue Name. . . . . : NT1.LQ1
Remote Queue Manager Name. : NT1.QM1
Transmission Queue Name. . : VSE1.XQ1




Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read   PF5=Add  PF6=Update
                                       PF9=List PF12=Delete
```

Figure 27. Remote queue definition

On this screen, the data entry fields are:

Object Name
Filled in from the previous screen.

Description lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Put Enabled
A toggle that enables or disables MQPUT operations against this queue.

Get Enabled
A toggle that enables or disables MQGET operations against this queue. For remote queue definitions, the Get Enabled queue parameter is ignored because it is not possible to issue an MQGET call against a remote queue.

Remote Queue Name
The queue name on the remote MQSeries system to which the definition in progress will refer.

Remote Queue Manager Name
The name of the remote MQSeries system queue manager on which the

remote queue is defined as a local queue. This name must be defined as a local transmission queue unless the Transmission Queue Name field is used.

Transmission Queue Name
The name of the local transmission queue to be used by MQSeries to convey messages to this remote queue. If the field is left blank, the remote queue manager name is required to map to a local transmission queue.

Note: Some other operating systems, with which you could be communicating, may be case sensitive. You should read the information in "Uppercase translation" on page 18 before devising a name for a queue, channel or queue manager.

Navigation through the screens is dependent upon the PF keys.

Create alias queue

Selecting "AQ" on the screen in Figure 24 on page 76 displays the screen shown in Figure 28.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
10:09:40               Queue Definition  Record                    CIC1
MQWMQUE           QM - VSE.TS.QM1                                   A002

                      Alias Queue Definition

Object Name. . . . . . . . : EMPLOYEE
Description line 1 . . . . : Alias for EMPLOYEE.DETAILS queue
Description line 2 . . . . : with PUT inhibited.

Put Enabled  . . . . . . . : N   Y=Yes, N=No
Get Enabled  . . . . . . . : Y   Y=Yes, N=No

ALIAS Queue Name . . . . . : EMPLOYEE.DETAILS









Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF12=Delete
```

Figure 28. Alias queue definition

On this screen, the data entry fields are:

Object Name
Filled in from the previous screen.

Description lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Put Enabled
A toggle that enables or disables MQPUT operations against this queue.

Get Enabled
A toggle that enables or disables MQGET operations against this queue.

Alias Queue Name
The name of another object already defined in the local configuration. This must be a local queue name. It cannot identify another alias.

Navigation through the screens is dependent upon the PF keys.

Create alias queue manager

Selecting "AM" on the screen in Figure 24 on page 76 displays the screen shown in Figure 29.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
10:13:10               Queue Definition  Record                     CIC1
MQWMQUE           QM - VSE.TS.QM1                                   A002

                      Alias Queue Manager Definition

Object Name. . . . . . . . : VSE.LOCAL1
Description line 1 . . . . : Alias Queue Manager
Description line 2 . . . . : for VSE.QM1

Alias Queue Manager Name . : VSE.QM1
Transmission Queue Name. . :









Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF12=Delete
```

Figure 29. Alias queue manager definition

On this screen, the definitions cannot be used in an MQCONN call; they may used for MQOPEN substitution only. The data entry fields are:

Object Name
Filled in from the previous screen.

Description lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Alias Queue Manager Name
The name of a known queue manager. This can be a local transmission queue name, a remote queue manager name, or the local queue manager name. It cannot identify another alias.

Transmission Queue Name
The name of the local transmission queue to be used by MQSeries to convey messages to this remote queue manager. If this field is left blank, the Alias Queue Manager Name field is required to map to a local transmission queue or to the local queue manager name.

Navigation through the screens is dependent upon the PF keys.

Create alias reply queue

Selecting "AR" on the screen in Figure 24 on page 76 displays the screen shown in Figure 30.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
10:20:24               Queue Definition  Record                     CIC1
MQWMQUE           QM - VSE.TS.QM1                                   A002

                      Alias Reply Queue Definition

Object Name. . . . . . . . : REPLYQ
Description line 1 . . . . : Alias reply definition
Description line 2 . . . . :

Alias Queue Name . . . . . : ANYQ
Alias Queue Manager Name . : VSE.TS.QM1









Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF12=Delete
```

Figure 30. Alias queue reply definition

On this screen, the definitions cannot be used in the MQOPEN call; they may only be used for reply queue name substitution with a MQPUT call. The data entry fields are:

Object Name
Filled in from the previous screen.

Description lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Alias Queue Name
The name of another object already defined in the local configuration. This can be a local queue name or a remote queue name. It cannot identify another alias.

Alias Queue Manager Name
The name of a known queue manager. This can be a local transmission queue name, a remote queue manager name, or the local queue manager name. It cannot identify another alias.

Navigation through the screens is dependent upon the PF keys.

Modifying and deleting queue definitions

Selecting 2 on the configuration menu also allows you to modify or delete queue definitions.

Note: You use the same primary screens for modifying, deleting, or adding a queue.

Selecting an existing queue definition

To modify or delete an existing queue definition, you must select the definition on which to work, and display it.

To do this, select option 2 on the "Configuration Main Menu" screen to display the "Queue Main Options" screen (see Figure 24 on page 76) and use either the PF4, or PF9, function key.

PF4 is the Read key, and you use it to bring a specific queue definition to the screen as follows:

1. Enter the name of the desired queue in the Object Name field.
2. Press PF4 or Enter.
3. MQSeries reads and displays the selected queue definition.

PF9 is the **LIST** key, and you use it to bring a specific queue definition to the screen as follows:

1. Press PF9.
2. The MQSeries System displays a list of all defined queues (see Figure 31).
3. Select the desired queue by typing an "X" next to its name, or by placing the cursor on the appropriate object.
4. Press PF4 or Enter.
5. MQSeries reads and displays the selected queue definition.

```
07/06/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQBDTS
16:24:52                   Object List Screen                       CIC1
MQMMQUE                                                             A001


S Object                                          Type
  ANYQ                                            Local Queue
  EMPLOYEE                                        Alias Queue
  REPLYQ                                          Alias Reply
  SYSTEM.LOG                                      Local Queue
  VSE.LOCAL1                                      Alias Manager
  VSE1.NT1.RQ1                                    Remote Queue









Records found                - Select one object name.

PF2 = Options   PF3 = Quit  PF4/Enter = Read
                PF7 = Backward  PF8 = Forward
```

Figure 31. Object list screen

Modifying an existing queue definition

When you have displayed the required queue definition, as described in "Selecting an existing queue definition," you can modify any field in the definition. This may involve multiple screens to include all fields of the queue definition – see "Queue definitions" on page 76.

When you have made the changes you need, update the screen using the PF6 (UPDATE) function key.

Deleting an existing queue definition

When you have displayed the required queue definition, as described in "Selecting an existing queue definition" on page 87, you can delete it using the PF12 (DELETE) function key.

You will be asked to confirm that you want to delete the queue definition. You must press the PF12 function key again to delete the queue.

Warning: Deleting a queue definition also deletes the queue's message data. If the message data is important, a queue should be emptied by normal application processing before it is deleted.

Channel definitions

Selecting 3 on the configuration menu allows you to maintain (add, modify, or delete) channel definitions for the local installation of the MQSeries System.

Note: The same screens are used to accomplish the three functions of adding, modifying, or deleting a channel definition; the required action being selected through the function keys. The following sections present the screens that you see if you are adding a new channel definition.

"Modifying and deleting channel definitions" on page 91 explains how you modify or delete a queue.

The screen shown in Figure 32 is displayed to create a channel definition:

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
 15:19:47                    Channel Record           DISPLAY         CIC1
 MQMMCHN                                                              A000
 Channel  : VSE7.TO.VSE8
  Desc. . :
  Protocol: T (L/T)     Type : S (Sender/Receiver/svrConn)  Enabled : Y

 Sender
  Remote TCP/IP port . . . . : 01414      LU62 Allocation retry num : 00000000
  Get retry number . . . . . : 00000003   LU62 delay fast (secs). . : 00000000
  Get retry delay (secs) . . : 00000015   LU62 delay slow (secs). . : 00000000
  Convert msgs(Y/N). . . . . : N
  Transmission queue name. . : VSE7.VSE8.XQ5
  TP name. . :

 Sender/Receiver
  Connection : VSE8HOST
  Max Messages per Batch . . : 000050     Message Sequence Wrap . . : 999999
  Max Message Size   . . . . : 0005000    Dead letter store(Y/N)  . : N
  Max Transmission Size  . . : 032766     Split Msg(Y/N)  . . . . . : N
  Max TCP/IP Wait  . . . . . : 000300

 Channel record displayed.
 F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

Figure 32. Channel record

On this screen, the data entry fields are:

General

Channel name
The name of the channel to be defined.

Description
The description of the channel. This field is for documentation purposes only, and does not affect channel operation.

Protocol
The protocol being used by the selected channel, which can be LU 6.2, or TCP/IP.

Type S: A sender only channel.

R: A receiver only channel.

C: A client channel.

Requester Channels are not supported for IBM MQSeries for VSE/ESA.

Enable
Enable the channel for communications at initialization time.

Sender

The following parameters are relevant to all sender channels, however, parameters prefixed "LU62" are relevant to LU 6.2 channels only. Parameters relevant to client (SVRCONN) channels are the general parameters described above.

Remote TCP/IP Port
The port number that MQSeries uses for accepting TCP/IP connection requests. For the following type of channels the port number is
RECEIVER
Not required
CLIENT
Not required
SENDER
That of the remote listener

The number reserved for MQSeries is 1414, although any unreserved number can be used.

Get retry number
The number of Get retries when queue is empty.

Get retry delay
The time between retries (in seconds).

Convert Msgs
A field that identifies whether message data is converted before it is sent to a remote queue manager. To convert message data, set this field to Y. Data is converted to the code page of the remote host only if the code page is supported.

Transmission Queue Name
The name of the transmission queue. Required for the sender, optional for the receiver.

TP Name
A 64-character field identifying the remote task ID of the receiver on the remote CICS region, or a TPNAME on the remote system (for example, AMQCRS6A). This is required by the sender for LU 6.2 use only. This field is not recognized for TCP/IP.

Note: Although the TPNAME can be up to 64 bytes in other MQSeries products, on MQSeries for VSE/ESA it can have a maximum of 4 bytes only.

LU62 Allocation Retry Num
Number of allocation retries when not successful. This parameter is for LU 6.2 channels only. TCP/IP channels can set this parameter to zero.

LU62 Delay fast
Time between retries (in seconds). This parameter is for LU 6.2 channels only. TCP/IP channels can set this parameter to zero.

LU62 Delay slow
Time between retries (in seconds) after fast number of retries has been depleted. This parameter is for LU 6.2 channels only. TCP/IP channels can set this parameter to zero.

Sender/Receiver

The following parameters are relevant to both sender and receiver channels.

Connection
The name of the LU 6.2 connection, or for TCP/IP sender channels, the remote host name or IP address.

Note that for TCP/IP channels, the connection name does not include a port number as it may do on other MQ platforms. The port number is configured as a separate channel parameter.

Max Messages per Batch
The mutually accepted maximum number of messages per batch to be transmitted.

Message Sequence Wrap
The mutually agreed maximum messages count before the count sequence starts over.

Dead Letter Store
A field that identifies whether inbound messages are written to the dead letter queue whenever an inbound message cannot be written to its intended target queue. The dead letter queue is identified in your Global System Definition.

Max Transmission Size
The mutually accepted maximum number of bytes per transmission.

Max Message Size
The mutually accepted maximum number of bytes per message. The Maximum Message Size must be bigger than the application message size, plus approximately 1,000 bytes for the MQSeries header.

Split Msg
This **must** be set to `Y` if messages longer than the maximum transmission size for the channel are to be sent across the channel.

Max TCP/IP Wait
The maximum number of second that a Message Channel Agent (MCA) should wait to receive TCP/IP data before terminating the connection with an error. See “Bullet-proof channels” on page 57 for more information.

Modifying and deleting channel definitions

Selecting 3 on the configuration menu also allows you to modify or delete channel definitions.

Note: You use the same primary screens for modifying, deleting, or adding a channel.

Selecting an existing channel definition

To modify or delete an existing channel definition, you must select the definition on which to work, and display it.

To do this, select option 3 on the "Configuration Main Menu" screen to display the "Channel Record" screen (see Figure 32 on page 88), and use either the PF4 or PF9 function key.

PF4 is the Read key, and you use it to bring a specific channel definition to the screen as follows:

1. Enter the name of the desired channel in the Channel Name field.
2. Press PF4 or Enter.
3. MQSeries reads and displays the selected channel definition.

PF9 is the List key, and you use it to bring a specific channel definition to the screen as follows:

1. Press PF9.
2. MQSeries displays a list of all defined channels (see Figure 33).
3. Select the desired channel by typing an "S" next to the channel name.
4. Press PF4 or Enter.
5. MQSeries reads and displays the selected channel definition.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
11:10:12                    Channel List                            CIC1
MQWMCHN                                                             A002
S CHANNEL NAME         TYPE STATUS   LAST MSN   CHECKPOINT
  MQTS.TO.MQ23           S  ENABLE           2  2002/12/23 10:49:41  TCP
  MQ23.TO.MQTS           R  ENABLE           1  2002/12/23 10:49:41  TCP
  MVSA_TO_VSEA           R  ENABLE           0  2002/12/23 10:49:41




ENTER 'S' to select Channel
F2=Return  PF3=Quit  PF4=Read  F7=Backward  PF8=Forward
```

Figure 33. Channel list

On this screen, the display fields are:

Channel Name
The names of all channels.

Type Type is sender (S), receiver (R), or client (C).

Status Channel may be enabled (ENABLE) or disabled (DISABL).

Last MSN
The last checkpointed message sequence number of the channel.

Checkpoint
The time of the last checkpoint.

Modifying an existing channel definition

When you have displayed the required channel definition, as described in "Selecting an existing channel definition" on page 91, you can modify any field in the definition.

When you have made the changes you need, update the screen using the PF6 (Update) function key.

Deleting an existing channel definition

When you have displayed the required channel definition, as described in "Selecting an existing channel definition" on page 91, you can delete it using the PF12 (Delete) function key.

You will be asked to confirm that you want to delete the channel definition. You must press the PF12 function key again to delete the channel.

Setting channel SSL parameters

If a channel requires secure sockets layer (SSL) services, then press PF10 (SSL) to display the Channel SSL Parameters screen:

```
11/22/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
14:34:00                  Channel SSL Parameters                    CIC1
MQMMCHN                                                             A001

     Channel Name: VSE1.TCP.AIX2          Type: S

     SSL Cipher Specification. : 0A       (2 character code)
     SSL Client Authentication : O        (Required or Optional)

     SSL Peer Attributes:
     > CN=IBM*                                                       <
     >                                                               <
     >                                                               <
     >                                                               <









SSL channel parameters displayed.
    PF2 = Return   PF3 = Quit     PF4 = Read      PF6 = Update
```

Figure 34. Channel SSL parameters

On this screen, the data entry fields are:

SSL Cipher Specification
The SSL Cipher Specification identifies a 2-character code for any of the supported SSL Version 3.0 ciphers. At the time of publication, these include:
01 for NULL MD5
02 for NULL SHA
08 for DES40 SHA for Export
09 for DES SHA for US
0A for Triple DES SHA for US
62 for RSA_EXPORT1024_DESCBC_SHA

It should be noted that the bit-size of an RSA private key can affect the list of valid cipher specifications. For more information, refer to relevant SSL product documentation.

It is expected that the list of supported ciphers will expand over time. Consequently, this field is not validated. Any non-blank value is accepted as a valid configuration value. However, if an unsupported code is used, the channel will fail on any attempted use. For this reason, SSL services documentation relevant to your current environment should be checked prior to setting this field.

This field, and this field alone, determines whether or not the channel will apply SSL services. In other words, the other SSL parameters in the channel configuration are ignored if this field is not set (that is, if it is left blank). Consequently, this field can be used to activate and deactivate SSL services on a channel without requiring the other parameters being reset.

SSL Client Authentication
SSL client authentication can be either

R Required
or
O Optional

If required, MQSeries checks that the remote SSL partner (MQSeries message channel agent or MQ client) provided a X.509 PKI certificate during SSL initial negotiation. If the remote partner fails to send a certificate, and one is required, the channel is terminated. If client authentication is optional, MQSeries ignores whether or not a certificate was received.

SSL Peer Attributes
SSL peer attributes is a 256-character case-sensitive field that can be used to ensure a remote partner's certificate contains identifiable attributes. This requires that the remote partner provided a certificate during SSL initial negotiation. If the remote partner fails to provide a certificate, then any check against the SSL Peer Attributes field will fail, and the channel will be terminated. The SSL Peer Attributes field expects a value (if any) in the form:

```
key=value,key=value, etc.
```

where *key* is one of the supported keywords (see below), the equal sign (=) is mandatory, and *value* is a value relative to the keyword that is expected to match the remote partner's certificate. Supported keywords include:

CN	Common name
C	Country
ST	State or province
L	Locality
O	Organization
OU	Organization Unit
SERIAL	Serial number

For example:

```
C=US,O=IBM
```

In this example, the Country field of the remote partner's X.509 PKI certificate must be "US", and the Organization field must be "IBM". Note that values can include imbedded blanks and wild cards (*). Values with embedded blanks must be enclosed in double quotes ("). If used, only one wild card can be used in each value, and only at the end of that value. For example:

```
C=US,O="IBM GSA *",CN=www.ibm.*
```

Following SSL negotiation, MQSeries checks that the remote partner's X.509 PKI certificate matches the SSL peer attributes specified by this field. If they match, channel activity proceeds with SSL services enabled. If they do not match, the channel is terminated.

Setting channel exit parameters

If the channel requires exit processing during its operation, Then press PF11 (Ext) to display the Channel Exit Settings screen:

```
11/12/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
14:46:55                  Channel Exit Settings                     CIC1
MQMMCHN                                                             A005

Channel name . . . : VSE1.TCP.AIX2
Channel type . . . : Sender

Send exit name . . :
Send exit data . . :

Receive exit name. :
Receive exit data. :

Security exit name :
Security exit data :

Message exit name. :
Message exit data. :




Channel exit settings displayed.
F2=Return  PF3=Quit  PF4=Read  F6=Update
```

Figure 35. Channel Exit settings

On this screen, the Channel name and type are display-only fields. The data entry fields are:

Send exit name
The name of the send exit program that is invoked prior to the transmission of data across an active channel.

Send exit data
Optional data that is passed to the send exit program on each invocation.

Receive exit name
The name of the receive exit program that is invoked immediately following the receipt of data across an active channel.

Receive exit data
Optional data that is passed to the receive exit program on each invocation.

Security exit name
The name of the security exit program that is invoked during the initial negotiation of the channel.

Security exit data
Optional data that is passed to the security exit program on each invocation.

Message exit name
The name of the message exit program that is invoked following a message being retrieved from a queue and prior to a message being placed on a queue during normal channel operation. Message exits are not valid for client (SVRCONN) channels, since these do not involve the transmission of MQ messages.

Message exit data
Optional data that is passed to the message exit program on each invocation.

In the case of exit names, these must be programs defined to the CICS region. Exit names can be 1-8 characters. Exit data can be 1-32 characters.

For more information on channel exits, see “Channel exits” on page 43.

Code page definitions

Selecting 4 on the configuration menu allows you to define global parameters and maintain a set of user-defined code pages for data conversion.

Figure 36 shows the screen that is displayed when you select 4 from the configuration menu.

```
07/07/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQBDTS
09:58:32                   Code Page Definition                     CIC1
MQMMCPG                 Data Conversion Information                 A001




          Default ASCII code page....: 0850



          Default EBCDIC code page...: 1047



          Convert EBCDIC newline.....: L    L=Ascii NL, T=Table, I=ISO




Record initialized - New record.
 PF2 = Return         PF3 = Quit     PF4/ENTER = Read    PF5 = Add PF6 =Update
                                     PF9 = List CodePages
```

Figure 36. Data conversion definitions

To change values on this screen, use PF6 (Update). To add user-defined code pages, use PF5 (Add).

On this screen, the data entry fields are:

Default ASCII code page
Code page used for default conversion when a code page that represents an ASCII code page fails normal conversion.

The default data conversion process is used if data conversion fails and the default ASCII code page is valid. Default conversion takes place only if the conversion is from ASCII to EBCDIC, or EBCDIC to ASCII. Otherwise, data is passed without conversion.

A value of 0 means no default conversion.

Default EBCDIC code page
Code page used for default conversion when a code page that represents an EBCDIC code page fails normal conversion.

The default data conversion process is used if data conversion fails and the default EBCDIC code page is valid. Default conversion takes place only if the conversion is from ASCII to EBCDIC, or EBCDIC to ASCII. Otherwise, data is passed without conversion.

A value of 0 means no default conversion.

Convert EBCDIC newline
This value is used only when converting EBCDIC to ASCII. Valid values are:

L: EBCDIC NL is converted to ASCII LF. This is the default behavior and how all V5.0 MQSeries products behave.

T: EBCDIC NL is converted to whatever value is specified in the supplied conversion table.

I: EBCDIC NL is converted to whatever value is specified inthe supplied conversion table. If the source is an ISO code page, the conversion is the same as L (NL to LF).

See the *MQSeries Application Programming Reference*, SC33-1673, for more details regarding conversion of EBCDIC newline characters.

Create a user-defined code page

Selecting option 4 on the configuration menu also allows you to create user-defined code pages.

To add a new user-defined code page, select option 4 on the configuration menu to display the Code Page Definition screen (see Figure 36 on page 96), then press PF5.

The screen shown in Figure 37 is displayed.

```
07/07/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQBDTS
10:42:50                 User Code Page Definition                  CIC1
MQMMCPG                                                             A001




 Code Page Number . . . . . : 1234
 Description Line 1 . . . . : Special ASCII code page
 Description Line 2 . . . . :

 Type . . . . . : S      S=SBCS, D=DBCS, M=MIXED
 Encoding . . . . . : A      E=EBCDIC, A=ASCII, I=ISO,
                             U=UCS2, T=UTF, C=EUC



Record being added   - Press ADD key again.

PF2=Data Conv   PF3 = Quit PF4/ENTER = Read   PF5 = Add          PF6 = Update
                PF9 = List                                      PF12= Delete
```

Figure 37. User code page definition

On this screen, the data entry fields are:

Code Page Number
Four digit number that uniquely identifies the user-defined code page. You cannot redefine a system-defined code page that already exists in LE/VSE.

Description lines 1 & 2
Text fields for operator use only. They may each be up to 32 characters.

Type S: Single Byte Character Set (SBCS)

D: Double Byte Character Set (DBCS)

M: Mixed SBCS/DBCS

Encoding

E: EBCDIC

A: ASCII

I: ISO

U: UCS-2 (Unicode)

T: UTF-8 (Unicode)

C: EUC

For more information on these types and encodings, see the IBM manual *Character Data Representation Architecture*, SC09-1390.

For a user code page to work, the underlying LE/VSE code converter must exist as a CICS phase. MQSeries for VSE/ESA uses LE/VSE for application message data conversion. See Chapter 7, “Message data conversion,” on page 163 for more details on message data conversion.

Modifying and deleting user-defined code pages

Selecting option 4 on the configuration menu also allows you to modify or delete user code page definitions.

Note: You use the same primary screens for both modifying and deleting a user code page definition.

Additionally, you can use the PF9 function key (List) from either the Code Page Definition screen, or the User Code Page Definition screen.

PF9 displays the code page Object List screen shown in Figure 38 on page 99.

```
07/07/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQBTDS
04:14:06                   Object List Screen                       CIC1
MQMMCPG                                                             A001


S Object                                            TYPE
  1123                                              SBCS Codepage
  1208                                              Mixed Codepage
  1308                                              SBCS Codepage
  3254                                              SBCS Codepage
  3722                                              SBCS Codepage
  4355                                              Mixed Codepage
  4545                                              SBCS Codepage
  4567                                              SBCS Codepage
  5657                                              SBCS Codepage
  6775                                              SBCS Codepage
...More



Records Found - Select one Object Name

PF2 =Data Conv  PF3 = Quit  PF4/Enter = Read
                PF7 = Backward  PF8 = Forward
```

Figure 38. Code page object list screen

Modifying an existing code page definition

From the displayed list, type an S next to the code page number you want to modify, or position the cursor on the entry, then press Enter.

When the required code page definition is displayed, you can edit modifiable fields and update the entry using PF6.

Deleting an existing code page definition.

From the displayed list, type an S next to the code page number you want to delete, or position the cursor on the entry, then press Enter.

When the required code page definition is displayed, use PF12 to delete the entry.

You are asked to confirm that you want to delete the definition. You must press the PF12 key again to delete the code page definition.

Global system definition display

Selecting 5 on the main menu allows you to view the attributes defined for the local queue manager, and all system-wide parameters, through the screen shown in Figure 39 on page 100, which is display only:

Global definition display

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQ212
12:37:03                  Global System Definition                   CIC1
MQWMSYS                 Queue Manager Information                    A002
Queue Manager . . . . . .: VSE.TS.QM1
Description Line 1. . . . .:
Description Line 2. . . . .:
                        Queue System Values
Maximum Number of Tasks . .: 00001000       System Wait Interval : 00000030
Maximum Concurrent Queues .: 00001000       Max. Recovery Tasks  : 0000
Allow TDQ Write on Errors  : Y    CSMT      Allow Internal Dump  : Y
                        Queue Maximum Values
Maximum Q Depth . . . . . .: 00640000       Maximum Global Locks.: 00001000
Maximum Message Size. . . .: 00004096       Maximum Local Locks .: 00001000
Maximum Single Q Access . .: 00000100
                        Global QUEUE /File Names
Local Code Page . . : 01047
Configuration File. : MQFCNFG
LOG Queue Name. . . : SYSTEM.LOG
Dead Letter Name. . : SYSTEM.DEAD.LETTER.QUEUE
Monitor Queue Name. : SYSTEM.MONITOR

Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read           PF9=Comms  PF10=Log   PF11=Event
```

Figure 39. Global system definition display

To return to the configuration main menu, press the PF2 key. To show display-only screens for system-wide communications settings or log and trace settings, press PF9 (Comms), PF10 (Log), and PF11 (Event) respectively.

Queue definition display

Selecting 6 on the main menu allows you to view existing queue definitions.

Note: This function allows you to see the queue definition, not the current queue status. To see the current queue information, see "Monitor queues" on page 110.

This operation is identical to the modify queue and delete queue operations (as described in "Modifying and deleting queue definitions" on page 86), except that the maintenance function keys PF5 (Add), PF6 (Update), and PF12 (Delete), are not available.

Channel definition display

Selecting 7 on the main menu allows you to view existing channel definitions.

Note: This function allows you to see the channel definition, not the current channel status. To see the current channel information, see "Monitor channel" on page 112.

This operation is identical to the modify channel and delete channel operations (as described in "Modifying and deleting channel definitions" on page 91), except that the maintenance function keys PF6 (Update) and PF12 (Delete), are not available.

Code page definition display

Selecting 8 on the configuration menu allows you to view existing code page defaults and user-defined code page definitions.

This operation is identical to the modify and delete code page operations, as available using option 4 of the configuration menu, except that the maintenance function keys PF6 (Update) and PF12 (Delete) are not available.

Operations functions

Selecting option 2 (Operations) from the master terminal main menu (see Figure 18 on page 64) displays the following screen:

```
12/27/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
13:02:40              *** Operations Main Menu ***                  CIC1
MQWMOPR                                                             A002
                             SYSTEM IS ACTIVE

                          1. Start / Stop Queue(s)

                          2. Open / Close Channel(s)

                          3. Reset Message Sequence Number

                          4. Initialization / Shutdown of System

                          5. Maintain Queue Message Records

                 Option:









    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Select   PF2=Return   PF3=Quit
```

Figure 40. Operations main menu

On this screen, selections correspond to available operator control functions.

Start/Stop queue

Selecting 1 on the operations menu allows you to start or stop processing for a queue.

This differs from setting the queue's Get Enabled or Put Enabled option to "No" in that the Start and Stop functions apply universally to all processes attempting to access a local queue.

The Get Enabled and Put Enabled functions can be selectively applied to aliases and remote queue definitions.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:07:54               Start / Stop Queue                           CIC1
MQWMSS                                                              A002

                            System Information
          System Status    : SYSTEM IS ACTIVE
          Queue Status     : Queuing System is active.
          Channel Status   : Channel System is active.
          Monitor Status   : Monitor is not active.
                           Single Queue Request
          Queue Name       :
          Function         :         S=Start, X=Stop,  R=Refresh from Config
          Mode             :         I=Inbound, O=Outbound, B=Both

          INBOUND Status   :
          OUTBOUND Status  :

                           Queuing System Request
          Function         :         S=Start, X=Stop, or M=Monitor


Please enter a Queue name.
Enter=Display  PF2=Return  PF3=Exit  PF6=Update
```

Figure 41. Start / Stop queue control screen

On the Start / Stop Queue screen shown in Figure 41, the fields are:

System Information

System Status
Reflects the status of the system. This is normally ACTIVE, unless the system has not been initialized, or has been shut down. When this occurs, the field reads SYSTEM IS SHUTDOWN.

Queue Status
Reflects the status of the queuing system. This is normally ACTIVE, unless the system has not yet been initialized or all the queues have been stopped. When this occurs, the field reads QUEUING SYSTEM IS STOPPED.

Channel Status
Reflects the status of the channels. This is normally ACTIVE, unless the system has not yet been initialized or all the channels have been closed. When this occurs, the field reads CHANNEL SYSTEM IS CLOSED.

Monitor Status
Reflects the status of the system monitor.

Single Queue Request

Queue Name
The name of a specific queue to start or stop

Function
The function to be performed, as follows:

S is to start:
- A stopped local queue
- The associated trigger mechanism
- Receiving messages if the channel is open

X is to stop a local queue and make it unavailable.

R is to refresh the run-time information for this queue from the configuration file, which was updated either by checkpoint requests or MQMT queue configuration. The configuration file (MQFCNFG) contains definitions of the queue manager, channels and queues.

Note: MQSeries configuration is dynamic and should not need to be refreshed during normal operation. If changes have been made to a queue and these changes do not appear to have been recognized by the queue manager, use the (R)efresh option to manually refresh the queue's definition.

Mode The queue process to be operated on, as indicated on screen.

INBOUND Status
Reflects the status of the specified queue. This is normally ACTIVE or IDLE unless the queue inbound has been stopped. If the queue is stopped, DISABLED is also displayed.

OUTBOUND Status
Reflects the status of the specified queue. This is normally ACTIVE or IDLE unless the queue outbound has been stopped. If the queue is stopped, DISABLED is also displayed.

Queuing System Request

Function
The function to be performed, as follows:

S is to start the system queue manager without affecting system resources.

X is to stop the system queue manager without affecting system resources.

Note: MQSeries configuration is dynamic and should not need to be refreshed during normal operation. If changes have been made to a queue and these changes do not appear to have been recognized by the queue manager, use the (R)efresh option to manually refresh the queue's definition.

M toggles the monitor flag. This flag is used to log application requests and their results to the system monitor queue.

Notes:

1. Only local queues can be stopped or started. In order to stop or start a queue that is not local, the queue definition must be updated in the Put-Enabled or Get-Enabled fields.
2. When a local queue is started, any associated triggers will also be started, if the queue depth reflects that messages are present.

   This does not happen when a "Queuing System Request" function is performed. Additionally, any queues that were stopped before the "Queuing System Request" stop function was performed, remain stopped when the corresponding start function is performed.

   Use the Monitor Queue function to check which local queues are stopped.
3. If the queue definition specifies a trigger and a sender channel, then starting a queue triggers the sender program to activate the channel and transmit messages.

Open / Close channel

Selecting 2 on the operations main menu (see Figure 40 on page 101) allows you to open or close communications on an existing channel.

Note: Opening or closing a channel is not the same as starting or stopping the MCA process. See "Communications process" on page 117 for further information.

Figure 42 shows the first screen to be displayed:

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:21:12               Open / Close Channel                         CIC1
MQWMSC                                                              A002

                           System Information
           System Status    : SYSTEM IS ACTIVE
           Queue Status     : Queuing System is active.
           Channel System   : Channel System is active.
                        Single Channel Request
           Channel Name     :

           Function         :        O=Open , C=Close

           Status           :



                            Channel System Request
           Function         :        O=Open , C=Close


Enter required information.
Enter=Refresh  PF2=Return  PF3=Exit  PF6=Update
```

Figure 42. Open / Close Channel

On this screen, the fields are:

System Information

System Status
Reflects the status of the system. This is normally ACTIVE, unless the system has not been initialized, or has been shut down. When this occurs, the field reads SYSTEM IS SHUTDOWN.

Queue Status
Reflects the status of the queuing system. This is normally ACTIVE, unless the system has not yet been initialized or all the queues have been stopped. When this occurs, the field reads QUEUING SYSTEM IS STOPPED.

Channel System
Reflects the status of the channels. This is normally ACTIVE, unless the system has not yet been initialized or all the channels have been closed. When this occurs, the field reads CHANNEL SYSTEM IS CLOSED.

Single Channel Request

Channel Name
The name of a specific channel to start or stop

Function
The function to be performed:

- O is to open a closed channel.
- C is to close an open channel.

Status Reflects the status of the specified channel. This is normally ACTIVE or IDLE unless the channel has been stopped. In this situation DISABLED is also displayed.

Channel System Request

Function
This either opens or closes the channel system.

Note: Opening a channel does not cause a trigger to activate. However, starting the channel's transmission queue does activate a trigger; see Note 3 on page 103.

Reset message sequence number

Selecting 3 on the operations main menu (see Figure 40 on page 101) allows you to reset the message sequence numbers on an existing channel by displaying the screen shown in Figure 43:

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:26:47               Reset Channel Message Sequence               CIC1
MQWMMSN                                                             A002

                          System Information
            System Status    : SYSTEM IS ACTIVE
            Queue Status     : QUEUING SYSTEM IS ACTIVE
            Channel Status   : CHANNEL SYSTEM IS ACTIVE
                          Reset Channel Info
            Channel Name     : MQTS.TO.MQ23


            Status           : IDLE

            Current Next-MSN : 000000001
            New     Next-MSN :




Information displayed.
Enter=Refresh  PF2=Return  PF3=Exit  PF6=Update
```

Figure 43. Reset channel message sequence

On this screen, the fields are:

System Information

System Status
Reflects the status of the system. This is normally ACTIVE, unless the system has not been initialized, or has been shut down. When this occurs, the field reads SYSTEM IS SHUTDOWN.

Queue Status
Reflects the status of the queuing system. This is normally ACTIVE, unless

the system has not yet been initialized or all the queues have been stopped. When this occurs, the field reads QUEUING SYSTEM IS STOPPED.

Channel Status
Reflects the status of the channels. This is normally ACTIVE, unless the system has not yet been initialized or all the channels have been closed. When this occurs, the field reads CHANNEL SYSTEM IS CLOSED.

Reset Channel Info

Channel Name
The name of a specific channel.

Status Reflects the status of the specified channel. This is normally ACTIVE or IDLE unless the channel has been stopped. In this situation DISABLED is also displayed.

Current Next-MSN
Displays the message sequence number to be used next.

New Next-MSN
Operator-entered value for message sequence number to be used next.

Note: In order for a channel message sequence number to be reset, the channel must be stopped.

Initialization of system

Selecting 4 on the operations menu allows you to initialize the queuing system by displaying the screen shown in Figure 44.

```
12/27/2002           IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:29:47          Initialization / Shutdown of System                CIC1
MQWMSI                                                               A002

                         System Information
           System Status    : SYSTEM IS ACTIVE
           Queue Status     : QUEUING SYSTEM IS ACTIVE
           Channel Status   : CHANNEL SYSTEM IS ACTIVE


           Function         :          I=Initialize, X=Shutdown


           Returned Results :


           System initialized on   12/23/2002 at 10:49:40



Please enter one of the options listed.

Enter=Refresh  PF2=Return  PF3=Exit  PF6=Update
```

Figure 44. Initialization of system

On this screen, the fields are:

System Information

System Status
Reflects the status of the system. This is normally ACTIVE, unless the system has not been initialized, or has been shut down. When this occurs, the field reads SYSTEM IS SHUTDOWN.

Queue Status
Reflects the status of the queuing system. This is normally ACTIVE, unless the system has not yet been initialized or all the queues have been stopped. When this occurs, the field reads QUEUING SYSTEM IS STOPPED.

Channel Status
Reflects the status of the channels. This is normally ACTIVE, unless the system has not yet been initialized or all the channels have been closed. When this occurs, the field reads CHANNEL SYSTEM IS CLOSED.

Function
The function to be performed:

- I is to initialize the system. If the system is initialized, the queue manager is started and all channels and queues opened. Any trigger associated with queues just initialized is also activated if the queue depth is nonzero.
- X is to shut down the system. If the system is shut down, the queue manager is stopped and all the channels closed.

Returned Results
Pressing PF6 with an Initialize function (I) on this screen causes the static system configuration files to be loaded into the CICS/VSE dynamic storage. Any error messages or progress messages are displayed in this field.

Note: All outstanding queue maintenance requests must have completed before you perform an initialize or shutdown operation.

Queue maintenance

Selecting 5 on the operations menu allows you either to reset deleted records or physically delete records, by displaying the screen shown in Figure 45 on page 108.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:31:20            Maintain Queue Message Records                  CIC1
MQWMDEL                                                             A002
                            System Information
         System Status    : System is active.
         Queue  Status    : Queuing system is active.
         Channel System   : Channel system is active.
                          Queue Information
         Queue Name       :
         Function         :   A=Delete all
                              D=Delete to date/time exclusive
                              R=Reset from date/time inclusive
         Date (yyyymmdd)  :
         Time (hhmmss)    :

                          Results of Request
         Number Processed :
         Number of Bypass :
         New Last Read QSN:
         Process Time     :

Please enter a Queue name.
Enter=Refresh  PF2=Return  PF3=Exit  PF6=Update
                                          PF12=Retry
```

Figure 45. Maintain Queue Message Records

On this screen, the fields are:

System Information

System Status
Reflects the status of the system. This is normally ACTIVE, unless the system has not been initialized, or has been shut down. When this occurs, the field reads SYSTEM IS SHUTDOWN.

Queue Status
Reflects the status of the queuing system. This is normally ACTIVE, unless the system has not yet been initialized or all the queues have been stopped. When this occurs, the field reads QUEUING SYSTEM IS STOPPED.

Channel System
Reflects the status of the channels. This is normally ACTIVE, unless the system has not yet been initialized or all the channels have been closed. When this occurs, the field reads CHANNEL SYSTEM IS CLOSED.

Queue Information

Queue Name
The name of the local queue on which the function is to be performed.

Function
The function to be performed:

- D is to delete messages that have been logically deleted up to a specified "written" exclusive date and time. For example, given the date and time of 980227230000, specifying "D" deletes all records with a written time prior to 11:00:00 p.m.

  Note: Specifying D does not actually reclaim VSAM space, because record keys are always created in ascending sequence. You are

strongly recommended to read "VSAM file maintenance" on page 140 for information regarding the Delete All function in relation to VSAM files.

- A is to delete all records (logically deleted, or written) and reclaim VSAM space
- R is to reset all logically deleted records to written status from a specified "deleted" inclusive date and time. For example, given the date and time of 980227230000, specifying "R" resets all delivered messages with delivery time after 10:59:59 p.m.

Date The last date up to which the selected function is to be performed, if applicable.

Time The last time up to which the selected function is to be performed, if applicable.

Queue Information

Number Processed
Number of messages processed

Number of Bypass
Number of messages bypassed

New Last Read QSN
Last read queue sequence number

Process Time
Time to process the request

You use the PF6 (Update) function key to activate the requested function. The function itself is performed by another task, which signals the screen when it is complete. To display the signal, press the Enter key.

The PF12 (Retry) function key is used only if the delete task has ended before finishing the current request, and acts as a new PF6 request.

Notes:

1. A Delete or Reset Messages by Date and Time performs the requested function up to the selected date and time, but does not include records with that date and time.
2. If the queue is examined with the browse function, the put time of the last message to be reset should be the value for date and time.
3. The Delete All function purges all records, which include both logically deleted and nondeleted messages.

Once a queue maintenance task is in progress, the task flags the Queue Information entry and logically prevents any other task from accessing this queue. Any attempt to open this queue is rejected with the following message:

```
Queue has xxxx tasks attached. These must be purged.
```

The only action available at this point is to wait and try again later.

Monitoring functions

Selecting option 3 (Monitoring) from the master terminal main menu (see Figure 18 on page 64) displays the screen shown in Figure 46 on page 110.

```
12/27/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
13:34:06             *** Monitor Main Menu ***                      CIC1
MQWMMON                                                             A002
                           SYSTEM IS ACTIVE


                             1. Monitor Queue

                             2. Monitor Channel


                    Option:









Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Select  PF2=Return  PF3=Quit
```

Figure 46. Monitor Main Menu

Monitor queues

Selecting 1 on the monitor main menu allows you to monitor the current status of all existing local queues, using the screen shown in Figure 47.

```
12/27/2002            IBM MQSeries for VSE/ESA Version 2.1.2         TSMQ212
13:35:57                   Monitor Queues                            CIC1
MQWMMOQ                                                              A002
                           QUEUING SYSTEM IS ACTIVE
S QUEUE                         FILE     T INBOUND  OUTBOUND        LR   QDepth
  ANYQ                          MQFI001  N STOPPED  STOPPED          6       10
  SYSTEM.ADMIN.COMMAND.QUEUE    MQFI003  N IDLE     ACTIVE           5        0
  SYSTEM.DEAD.LETTER.QUEUE      MQFERR   N IDLE     IDLE             0        0
  SYSTEM.LOG                    MQFLOG   N IDLE     IDLE             0      134
  SYSTEM.MONITOR                MQFMON   N IDLE     IDLE             0        0
  VSE1.AIX1.XQ1                 MQFO002  N IDLE     IDLE             4       20
  VSE1.NT1.XQ1                  MQFO002  N IDLE     IDLE             0        2
  WWW.REQ.TYPE7                 MQFI003  Y IDLE     IDLE             3        0









Information displayed.
Enter=Refresh  PF2=Return  PF3=Exit  PF7=Back  PF8=Forward
                              PF9=All   Select or PF10=Detail
```

Figure 47. Monitor queues

This screen displays the current status of all local queues. The displayed fields are:

Queue
Name of the queue.

File CICS FCT DDNAME of a local queue definition.

T Queue type.
- N Normal local queue.
- Y Transmission local queue.

When PF9 (All) option is selected, additional type values may be displayed. They are:
- M Manager alias.
- A Queue alias.
- X Remote queue definition.

Inbound
Status of the inbound process.
- ACTIVE One or more users have the queue open for MQPUT operations.
- IDLE No user has the queue open for MQPUT operations.
- STOPPED Queue has been stopped.
- MAX At maximum QDepth.
- FULL No space.
- RECOVERY For dual queuing.

Outbound
Status of the outbound process.
- ACTIVE One or more users have the queue open for MQGET operations.
- IDLE No user has the queue open for MQGET operations.
- STOPPED Queue has been stopped.
- RECOVERY For dual queuing.

LR Last Read: relative record number of the last record on the queue that has been read and processed.

Note: MQSeries messages are logically, rather than physically, deleted from the queue file. **LR** tells you which physical record is prior to the first active record.

QDepth
Estimated queue depth: The approximate number of records currently on the queue, remaining to be processed.

Notes:

1. The estimated queue depth is based on all MQPUT requests and on syncpointed MQGET requests.
2. Pressing the PF9 (All) function key displays an entire list of all queues (local, remote, and alias) together with their associated reference.

   If you press this key again, the display lists local queues only.
3. Pressing the PF10 (Detail) function key displays detailed information for this local queue entry.

Monitor queues - detail

An individual queue can be monitored in more detail by placing the cursor anywhere on the line displaying the queue name and pressing PF10, or by placing any character (other than '/') in the S(election) field at the beginning of the line and pressing the ENTER key. Either approach activates the Monitor Queues Detail screen as shown in Figure 48 on page 112.

```
12/27/2002           IBM MQSeries for VSE/ESA Version 2.1.2         TSMQ212
13:41:05                  Monitor Queues                            CIC1
MQWMMOQ                                                             A002
                          QUEUING SYSTEM IS ACTIVE
                          DETAIL QUEUE INFORMATION
VSE1.AIX1.XQ1
 INBOUND:   STATUS I  ENABLED Y   OPEN Q         0
            CHECKPOINT:            TAKEN         0
 OUTBOUND:  STATUS I  ENABLED Y   OPEN Q         0
            CHECKPOINT:            TAKEN         0

 BOTH:     FIQ         5    LIQ        24   GETS         0    QDEPTH       20
 TRIGGER:  MAX         1   USED         0   TRAN       PROG. MQPSEND
           CID VSE1.TO.AIX1









Information displayed.
Enter=Refresh  PF2=Return  PF3=Exit  PF10=List
```

Figure 48. Monitor Queues - detail

Monitor channel

Selecting 2 on the monitor main menu (shown in Figure 46 on page 110) allows you to monitor the current status of existing local communications channels, using the screen shown in Figure 49.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQ212
13:44:35                 Monitor Channels                            CIC1
MQWMMOC                                                              A002
                        CHANNEL SYSTEM IS ACTIVE
S CHANNEL              TYPE       MSN       QSN      QUEUE
  AIX1.SNA.VSE1        RECV         0         0 I
  AIX2.TCP.VSE1        RECV        98        10 I WWW.REQ.TYPE7
  NT1.TCP.VSE1         RECV         1         1 I EMPLOYEE.PAY.TRANS089
  OS390A.SNA.VSE1      RECV         0         0 I
  VSE1.SNA.AIX1        SEND        54         2 I VSE1.AIX1.XQ1
  VSE1.SNA.OS390A      SEND         0         0 I
  VSE1.TCP.AIX2        SEND      2077       108 I VSE1.AIX2.XQ1
  VSE1.TCP.OS390C      SEND         0         0 I
  VSE1.TCP.VSE2        SEND         0         0 C
  VSE2.TCP.VSE1        RECV         2 23        I WWW.REQ.TYPE8
  ..More




Information displayed.
Enter=Refresh  PF2=Return  PF3=Exit
PF7=Scroll Back    PF8=Scroll Forward   Select or PF10=Detail
```

Figure 49. Monitor channel definitions

This screen displays the current status of local channels.

Channel
Name of the channel.

Type Sender (SEND) or receiver (RECV).

MSN Last channel message sequence number received or sent.

QSN Queue message sequence number, of the queue name displayed in the **Queue** field.

QUEUE
Name of the queue associated with the channel. This field is preceded by a single character identifying the channel status:

I IDLE

B BUSY

C CLOSED (for example, DISABLED)

Note: If this is a receiver channel, the QUEUE field displays the last queue name for which a message has been received.

Monitor channel - detail

Pressing PF10 (Detail) displays detailed information for a specific channel shown in Figure 50.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:48:59                   Monitor Channels                         CIC1
MQWMMOC                                                             A002
                          CHANNEL SYSTEM IS ACTIVE
                           DETAIL CHANNEL INFORMATION
MQTS.TO.MQ23
            COMMIT          MSN       QSN DATE/TIME
 Sender      BEFORE           0         0 20021126182613

             AFTER            2         3 20021126182613
                     MQTS.MQ23.XQ1









Information displayed.
Enter=Refresh  PF2=Return  PF3=Exit
                                                    PF10=List
```

Figure 50. Monitor channel definitions - detail

This screen displays the channel name, the channel type, and the name of the queue it accesses. The MSN, QSN and time stamp of the last commitment for the BEFORE and AFTER COMMIT fields are also shown.

Browse function

Selecting option 4 (Browse Queue Records) from the master terminal main menu (see Figure 18 on page 64) displays the screen shown in Figure 51 on page 114.

```
12/27/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
13:51:08                  Browse Queue Records                      CIC1
MQWDISP                     SYSTEM IS ACTIVE                        A002

    Object Name: ANYQ
    QSN Number : 00000001       LR-        6, LW-       16, DD-MQFI001
                      Queue Data Record
Record Status : Deleted         PUT date/time  : 20021126145946
Message Size  : 00000200        GET date/time  : 20021127092412
Queue line.
THIS IS A MESSAGE TEXT









Information displayed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Process  PF2=Return  PF3=Quit  PF4=Next  PF5=Prior
   PF7=Up      PF8=Down    PF9=Hex/Char  PF10=Txt/Head
```

Figure 51. Browse Queue Records

This screen shows the content of the message for the specified queue sequence number (QSN) of the chosen object name (queue name). Record status is shown as Written or Deleted along with the associated time stamps.

To browse the queue records, enter the local object name and the QSN of the message of interest. The message on the queue appears in the blank area of the screen, and the message can be manipulated using the function (PF) keys.

Pressing the PF9 (Hex/Char) key displays the message in hexadecimal or EBCDIC text code.

The PF10 (Txt/Head) key presents detailed MQSeries information for the selected record, and includes channel information if the queue is a transmission queue.

If you browse the System Log file, the PF12 (Help) key appears and can be used to display user action and system action for this message, provided that the system is active.

Notes:

1. If the file you are browsing is in the process of being updated by any other MQSeries tasks, this function waits until the completion of those tasks. There can be a delay in the response of the browse function.
2. The Browse utility also has the ability to browse from a queue when the queue manager is not active.

Administration via a web browser

Although MQSeries for VSE/ESA and its administration programs run in a CICS environment, administration of MQSeries objects can be performed via a web browser.

Administering MQSeries in this way takes advantage of the CICS Web Support (CWS) feature of CICS Transaction Server for VSE/ESA 1.1.1.

This section provides an overview of the CWS feature, describes the MQSeries modules that facilitate administration from a web browser, and explains how to use CWS with MQSeries.

CICS Web Support

CWS is a 2-tier connector solution based on HTTP/HTML. It facilitates connectivity between a web browser and VSE/ESA CICS TS applications. For MQSeries for VSE/ESA, this means connectivity between a web browser and the MQSeries administration programs.

The MQSeries administration programs are 3270-based CICS applications. CWS provides a 3270 bridge solution which allows access to 3270-based CICS applications without 3270 terminals. Administering MQSeries from a web browser exploits the 3270 bridge feature of CWS.

The operating environment and minimum prerequisites for CWS are:

- VSE/ESA 2.5.0
- CICS Transaction Server 1.1.1
- TCP/IP for VSE/ESA 1.4.0.
- LE/VSE 1.4.1

In addition, to administer MQSeries from a web browser, CWS must be correctly implemented, and the CICS/TS region hosting MQSeries must be properly configured for CWS use. To this end, you should refer to the following manuals:

- CICS Transaction Server for VSE/ESA Enhancement Guide (SC34-5763)
- CICS Transaction Server for VSE/ESA Internet Guide (SC34-5765)
- CICS Transaction Server for VSE/ESA CICS Web Support (SG24-5997-00)

Note that these manuals pertain to CICS Transaction Server for VSE/ESA V1.1.1. MQSeries administration via a web browser is not possible with CICS/VSE or CICS/TS releases prior to V1.1.1.

CWS MQSeries modules

MQSeries for VSE/ESA provides HTML source that corresponds to each of its administration screens. The source is provided in the MQSeries installation library in member MQHTML.Z.

In addition, MQSeries for VSE/ESA provides a 3270/HTML converter program called MQPCWS.

Each of these modules is described below.

HTML source file

The MQHTML.Z file contains HTML source generated by an assembly of the MQSeries MAP programs using the TYPE=TEMPLATE parameter of the DFHMSD macro. This is the standard method for generating HTML source from CICS Basic Mapping Support (BMS) programs.

The MQHTML.Z file contains a series of LIBRARIAN CATALOG statements followed by HTML source relevant to a particular administration screen. It is

intended to be used as input to the Librarian utility program (LIBR) to create individual HTML source members for each MQSeries administration screen.

The following JCL provides an example of how the MQHTML.Z file might be loaded into a sublibrary for CWS use:

```
* $$ JOB MQCWSLD,CLASS=0
* $$ LST CLASS=A,DISP=H
// JOB
// EXEC LIBR
ACC S=library.dfhdoc
* $$ SLI MEM=MQHTML.Z,S=PRD2.MQSERIES
/*
/&
* $$ EOJ
```

where `library.dfhdoc` should be replaced with a sublibrary name intended for CWS use. If MQSeries is installed in a sublibrary other than PRD2.MQSERIES, the `* $$ SLI` card should also be changed.

Once the HTML source has been loaded into an appropriate documents sublibrary (that is, a library configured for CWS), the HTML source is ready for use.

CWS converter program

Converter programs are optional programs used to support the operation of CWS. Essentially, a converter is an exit program called before and after the 3270/HTML data flow to and from a CICS TS program.

A converter must provide two functions:

- Decode is used before the CICS TS program is called. It can:
  - Use the data from the Web browser to build the communication area in the format expected by the CICS TS program.
  - Supply the lengths of the input and output data in the CICS TS program communication area.
  - Perform administrative tasks related to the response.
- Encode is used after the CICS TS program has been called. It can:
  - Use the data from the CICS TS program to build the HTTP response and HTTP response headers.
  - Perform administrative tasks related to the response.

On some browsers the web page layout generated from the provided HTML source may require improvement. For this reason, MQSeries for VSE/ESA provides a sample 3270/HTML converter program (MQPCWS) which is intended to improve the overall appearance of the MQSeries administration screens in a web browser.

The converter program, MQPCWS, is provided as a sample only, and is available in the MQSeries installation sublibrary as both an executable and a COBOL source file.

Using CWS with MQSeries

Once the CWS environment has been implemented and the MQSeries HTML files have been loaded into an appropriate CWS documents sublibrary, MQSeries can be administered from a web browser.

MQSeries does not need to be active to administer MQSeries from a browser since the activation of the system is itself an administration function that can be

performed from the administration screens. However, to administer MQSeries from a browser, TCP/IP services and the CICS TS region hosting MQSeries must be active.

MQSeries administration transactions are started from a browser by requesting an appropriate Universal Resource Locator (URL). For example:

```
http://n.n.n.n:pppp/cics/cwba/dfhwbtta/TTTT
```

where `n.n.n.n` is the IP address of VSE/ESA system running the CICS TS system that hosts MQSeries, pppp is the CICS listener port number for CWS, and TTTT is the MQSeries administration transaction identifier (for example MQMT).

Valid MQSeries transaction identifiers are listed in section "Master Terminal transactions" on page 64.

As already suggested, the appearance of the administration screens may be improved using the supplied CWS converter program, MQPCWS. A converter program can be introduced by changing the URL path. For example:

```
http://n.n.n.n:pppp/mqpcws/cwba/dfhwbtta/TTTT
```

Activating an MQSeries administration transaction this way will ensure that the the MQPCWS converter program is called to encode and decode the 3270/HTML data flow.

Communications process

MQSeries uses the Message Channel Agent (MCA) programs and TCP/IP Listener program for its communications.

The MCA process:

- Runs as a separate CICS task connected to the remote MQSeries using APPC or TCP/IP protocol.
- Starts automatically in response to other system activity, or when a message is placed on a transmission queue.
- Starts the MQSeries server when initial client requests are received.
- Can be stopped from the operations main menu.
- When processing a sender channel, ensures SSL services are activated if required.
- When processing a receiver channel, ensures SSL services are negotiated when required.

Select 2 from the operations main menu to control the channels. See "Open / Close channel" on page 104 for further information.

The MQSeries listener process:

- Establishes itself as a TCP/IP "listener" process by binding itself to a port number configured in the global system definition.
- Runs as a separate CICS task waiting for remote TCP/IP connection requests.
- Starts the receiver MCA when connection requests are received.
- Ends when MQSeries is shut down.

Message expiry

By default, messages placed on a queue have an unlimited 'lifetime'. This is the amount of time a message will stay on a queue before it is discarded by the queue manager, assuming it is not retrieved by an application.

Optionally, messages can be placed on a queue with a limited 'lifetime'. The amount of time a message will remain on a queue before it is discarded by the queue manager is called the message 'expiry'. A message's expiry is specified in the message descriptor (MQMD) data structure when the message is placed on a queue.

The message expiry is period of time expressed in tenths of a second, set by the application that puts the message. The message becomes eligible to be discarded if it has not been removed from the destination queue before this period of time elapses.

The value is decremented to reflect the time the message spends on the destination queue, and also on any intermediate transmission queues if the put is to a remote queue. It may also be decremented by message channel agents to reflect transmission times, if these are significant. Likewise, an application forwarding this message to another queue might decrement the value if necessary, if it has retained the message for a significant time. However, the expiration time is treated as approximate, and the value need not be decremented to reflect small time intervals.

When the message is retrieved by an application using the MQGET call, the Expiry field represents the amount of the original expiry time that still remains.

A message that has expired is never returned to an application (either by a browse or a non-browse MQGET call), so the value in the Expiry field of the message descriptor after a successful MQGET call is either greater than zero, or the special value MQEI_UNLIMITED.

If a message is put on a remote queue, the message may expire (and be discarded) whilst it is on an intermediate transmission queue, before the message reaches the destination queue.

A report is generated when an expired message is discarded, if the message specified one of the MQRO_EXPIRATION_* report options. If none of these options is specified, no such report is generated; the message is assumed to be no longer relevant after this time period (perhaps because a later message has superseded it).

Any other program that discards messages based on expiry time must also send an appropriate report message if one was requested.

Report messages are treated in the same way as ordinary messages; if the report message cannot be delivered to its destination queue (usually the queue specified by the ReplyToQ field in the message descriptor of the original message), the report message is placed on the dead-letter (undelivered-message) queue.

Notes:

1. If a message is put with an Expiry time of zero, the MQPUT or MQPUT1 call fails with reason code MQRC_EXPIRY_ERROR; no report message is generated in this case.

2. Since a message whose expiry time has elapsed may not actually be discarded until later, there may be messages on a queue that have passed their expiry time, and which are not therefore eligible for retrieval. These messages nevertheless count towards the number of messages on the queue for all purposes, including depth triggering.
3. An expiration report is generated, if requested, when the message is actually discarded, not when it becomes eligible for discarding.
4. Discarding of an expired message, and the generation of an expiration report if requested, are never part of the application's unit of work, even if the message was scheduled for discarding as a result of an MQGET call operating within a unit of work.
5. If a nearly-expired message is retrieved by an MQGET call within a unit of work, and the unit of work is subsequently backed out, the message may become eligible to be discarded before it can be retrieved again.
6. If a nearly-expired message is locked by an MQGET call with MQGMO_LOCK, the message may become eligible to be discarded before it can be retrieved by an MQGET call with MQGMO_MSG_UNDER_CURSOR; reason code MQRC_NO_MSG_UNDER_CURSOR is returned on this subsequent MQGET call if that happens.
7. When a request message with an expiry time greater than zero is retrieved, the application can take one of the following actions when it sends the reply message:
   - Copy the remaining expiry time from the request message to the reply message.
   - Set the expiry time in the reply message to an explicit value greater than zero.
   - Set the expiry time in the reply message to MQEI_UNLIMITED.

   The action to take depends on the design of the application suite. However, the default action for putting messages to a dead-letter (undelivered-message) queue should be to preserve the remaining expiry time of the message, and to continue to decrement it.
8. Expiry report messages are always generated with MQEI_UNLIMITED.
9. A message (normally on a transmission queue) which has a Format name of MQFMT_XMIT_Q_HEADER has a second message descriptor within the MQXQH. It therefore has two Expiry fields associated with it. The following additional points should be noted in this case:
   - When an application puts a message on a remote queue, the queue manager places the message initially on a local transmission queue, and prefixes the application message data with an MQXQH structure. The queue manager sets the values of the two Expiry fields to be the same as that specified by the application.
   - If an application puts a message directly on a local transmission queue, the message data must already begin with an MQXQH structure, and the format name must be MQFMT_XMIT_Q_HEADER (but the queue manager does not enforce this). In this case the application need not set the values of these two Expiry fields to be the same. (The queue manager does not check that the Expiry field within the MQXQH contains a valid value, or even that the message data is long enough to include it.)
   - When a message with a Format name of MQFMT_XMIT_Q_HEADER is retrieved from a queue (whether this is a normal or a transmission queue), the queue manager decrements both these Expiry fields with the time spent

waiting on the queue. No error is raised if the message data is not long enough to include the Expiry field in the MQXQH.

- The queue manager uses the Expiry field in the separate message descriptor (that is, not the one in the message descriptor embedded within the MQXQH structure) to test whether the message is eligible for discarding.
- If the initial values of the two Expiry fields were different, it is therefore possible for the Expiry time in the separate message descriptor when the message is retrieved to be greater than zero (so the message is not eligible for discarding), while the time according to the Expiry field in the MQXQH has elapsed. In this case the Expiry field in the MQXQH is set to zero.

Instrumentation events

In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. Such an event causes the queue manager or channel instance to put a special message, called an event message, on an event queue.

MQSeries instrumentation events provide information about errors, warnings, and other significant occurrences in a queue manager. You can, therefore, use these events to monitor the operation of queue managers.

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 you can retrieve by writing a suitable MQI application program that:
Gets the message from the queue.
Processes the message to extract the event data.

MQSeries instrumentation events may be categorized as follows:
Queue manager events
Channel events
Performance events

Queue manager 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 are notifications that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached.

Channel events are reported by channels as a result of conditions detected during their operation. For example, when a channel instance is stopped.

For each queue manager, each category of event has its own event queue. All events in that category result in an event message being put onto the same queue. Event queues are configurable using MQMT option 1.1, PF11. This activates the following screen:

```
07/22/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
10:05:48                  Global System Definition                  CIC1
MQMMSYS                        Event Settings                       A000

    Event queues
    Queue manager events : SYSTEM.ADMIN.QMGR.EVENT
    Channel events . . . : SYSTEM.ADMIN.CHANNEL.EVENT
    Performance events . : SYSTEM.ADMIN.PERFM.EVENT

    Qmgr events         Channel events      Performance events
    Inhibit  . . . : Y  Started  . . . : Y  Queue depth  . . : N
    Local  . . . . : Y  Stopped  . . . : Y  Service interval : N
    Remote . . . . : Y  Conversion err : Y
    Authority  . . : Y
    Start/Stop . . : Y




Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 52. Global System Definition

The default names for the event queues are as follows:

This event queue:	Contains messages from:
SYSTEM.ADMIN.QMGR.EVENT	Queue manager events
SYSTEM.ADMIN.CHANNEL.EVENT	Channel events
SYSTEM.ADMIN.PERFM.EVENT	Performance events

By incorporating these events into your own system management application, you can monitor the activities across many queue managers, across many different nodes, for multiple MQSeries applications. In particular, you can monitor all the nodes in your system from a single node (for those nodes that support MQSeries events).

Instrumentation events can be reported through a user-written reporting mechanism to an administration application that supports the presentation of the events to an operator.

Review section “Features” on page 8 for prerequisites for this feature.

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. The event messages for queue manager events are put on the SYSTEM.ADMIN.QMGR.EVENT queue. The following queue manager event types are supported:

- Inhibit
- Local
- Remote
- Authority
- Start/Stop

For each event type in this list, there is a queue manager attribute that enables or disables the event type. The conditions that give rise to the event (when enabled) include:

- An application issues an MQI call that fails. The reason code from the call is the same as the reason code in the event message.

  Note that a similar condition may occur during the internal operation of a queue manager, for example, when generating a report message. The reason code in an event message may match an MQI reason code, even thoug it is not associated with any application. Therefore you should not assume that, because an event message reason code looks like an MQI reason code, the event was necessarily caused by an unsuccessful MQI call from an application.
- A command is issued to a queue manager and the processing of this comman causes an event. For example: A queue manager is stopped or started. A command is issued where the associated user ID is not authorized for that command.

Inhibit events

Inhibit events indicate that an MQPUT or MQGET operation has been attempted against a queue, where the queue is inhibited for puts or gets respectively.

There are two types of Inhibit event:
Get Inhibited
Put Inhibited

Local events

Local events indicate that an application (or the queue manager) has not been able to access a local queue, or other local object. For example, when an application attempts to access an object that has not been defined.

There are three types of Local event:
Alias Base Queue Type Error
Unknown Alias Base Queue
Unknown Object Name

Remote events

Remote events indicate that an application (or the queue manager) cannot access a (remote) queue on another queue manager. For example, when the transmission queue to be used is not correctly defined.

Remote events include the following:
Remote Queue Name Error
Transmission Queue Type Error
Transmission Queue Usage Error
Unknown Default Transmission Queue
Unknown Remote Queue Manager
Unknown Transmission Queue

Authority events

Authority events indicate that an authorization violation has been detected. For example, an application attempts to open a queue for which it does not have the required authority, or a command is issued from a user ID that does not have the required authority.

There are three types of Authority event supported by MQSeries: for VSE/ESA:
Not Authorized (type 1) - Connection failures
Not Authorized (type 2) - Open failures
Not Authorized (type 4) - Command failures

Start and stop events

Start and stop events indicate that a queue manager has been started or has been requested to stop.

Start and Stop events include the following:
Queue Manager Active
Queue Manager Not Active

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. Channel events are generated:

- By a command to stop a channel.
- When a channel instance starts or stops. Note that client connections do not cause Channel Started or Channel Stop events.
- When a channel receives a conversion error warning when getting a message.

Channel event messages are put onto the SYSTEM.ADMIN.CHANNEL.EVENT queue, if it is available, and the queue manager is configured to generate channel events. Otherwise, they are ignored.

Channel events include the following:
Channel Started
Channel Stopped
Channel Conversion Error

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.

Performance events are related to conditions that can affect the performance of applications that use a specified queue.

The event type is returned in the command identifier field in the messag data.

Performance events are not generated for the event queues themselves. If a queue manager attempts to put a queue manager or a performance even message on an event queue and an error that would normally create an event is detected, another event is not created and no action is taken.

MQGET calls and MQPUT calls within a unit of work can cause performance related events to occur regardless of whether the unit of work is committed or backed out.

There are two types of performance event:

- Queue depth events, which include:
  Queue Depth High
  Queue Depth Low
  Queue Full
- Queue service interval events, which include:
  Queue Service Interval High
  Queue Service Interval OK

Performance events

MQSeries for VSE/ESA supports queue depth and service internal events for local queues, including transmissions queues.

As already stated, performance events are related to conditions that can affect the performance of applications that use a specified queue.

The scope of performance events is the queue, so that MQPUT calls and MQGET calls on one queue do not affect the generation of performance events on another queue.

Note: A message must be either put on, or removed from, a queue for any performance event to be generated.

The event data contains a reason code that identifies the cause of the event, a set of performance event statistics, and other data.

The event data in the event message contains information about the event for system management programs. For all performance events, the event data contains the names of the queue manager and the queue associated with the event. Also, the event data contains statistics related to the event.

You can use these statistics to analyze the behavior of a specified queue. The following table summarizes the event statistics. All the statistics refer to what has happened since the last time the statistics were reset.

Parameter	Description
TimeSinceReset	The elapsed time since last reset.
HighQDepth	Maximum queue depth since last reset.
MsgEnqCount	Messages put since last reset.
MsgDeqCount	Messages got since last reset.

Performance event statistics are reset when any of the following occur:
A performance event occurs
A queue manager stops and restarts

Queue depth events

In MQSeries applications, queues must not become full. If they do, applications can no longer put messages on the queue that they specify.

Although the message is not lost if this occurs, it can be a considerable inconvenience. The number of messages can build up on a queue if the messages are being put onto the queue faster than the applications that process them can take them off.

The solution to this problem depends on the particular circumstances, but may involve:
Diverting some messages to another queue.
Starting new applications to take more messages off the queue.
Stopping nonessential message traffic.
Increasing the queue depth to overcome a transient maximum.

Clearly, having advanced warning that problems may be on their way makes it easier to take preventive action. For this purpose, queue depth event are provided.

Queue depth events are related to the queue depth, that is, the number of messages on the queue. The types of queue depth events are:

- Queue Depth High events, which indicate that the queue depth has increased to a predefined threshold called the Queue Depth High limit.
- Queue Depth Low events, which indicate that the queue depth has decrease to a predefined threshold called the Queue Depth Low limit.
- Queue Full events, which indicate that the queue has reached its maximum depth, that is, the queue is full.

  A Queue Full Event is generated when an application attempts to put a message on a queue that has reached its maximum depth. Queue Depth High events give advance warning that a queue is filling up. This means that having received this event, the system administrator should take some preventive action. If this action is successful and the queue depth drop to a "safe" level, the queue manager can be configured to generate a Queue Depth Low event indicating an 'all clear' state.

Queue service interval events

Queue service interval events indicate whether a queue was "serviced" within a user-defined time interval called the service interval. Depending on the circumstances at your installation, you can use queue service interval events to monitor whether messages are being taken off queues quickly enough.

There are two types of queue service interval event:

- A Queue Service Interval OK event, which indicates that, following an MQPUT call or an MQGET call that leaves a non-empty queue, an MQGET call was performed within a user-defined time period, known as the service interval.
- A Queue Service Interval High event, which indicates that, following an MQGET or MQPUT call that leaves a non-empty queue, an MQPUT or an MQGET call was not performed within the user-defined service interval.

To enable both Queue Service Interval OK and Queue Service Interval High events you need to set the QServiceIntervalEvent control attribute to High. Queue Service Interval OK events are automatically enabled when a Queue Service Interval High event is generated. You do not need to enable Queue Service Interval OK events independently.

These events are mutually exclusive, which means that if one is enabled the other is disabled. However, both events can be simultaneously disabled.

In terms of queue service interval events, these are the possible outcomes:

- If the elapsed time between the put and get is less than or equal to the service interval:

  A Queue Service Interval OK event is generated, if queue service interval events are enabled.
- If the elapsed time between the put and get is greater than the service interval:

  A Queue Service Interval High event is generated, if queue service interval events are enabled

Service interval timer

Queue service interval events use an internal timer, called the service timer,

which is controlled by the queue manager. The service timer is used only if one or other of the queue service interval events is enabled.

The service timer measures the elapsed time between an MQPUT call to an empty queue or an MQGET call and the next put or get, provided the queue depth is nonzero between these two operations.

The service timer is always active (running), if the queue has messages on it (depth is nonzero) and a queue service interval event is enabled. If the queue becomes empty (queue depth zero), the timer is put into an OFF state, to be restarted on the next put.

The service timer is always reset after an MQGET call. It is also reset by an MQPUT call to an empty queue. However, it is not necessarily reset on a queue service interval event.

Following an MQGET call or an MQPUT call, the queue manager compares the elapsed time as measured by the service timer, with the user-defined service interval. The result of this comparison is that:

- An OK event is generated if the operation is an MQGET call and the elapsed time is less than or equal to the service interval, AND this event is enabled.
- A high event is generated if the elapsed time is greater than the service interval, AND this event is enabled.

Enabling and disabling events

All instrumentation events must be enabled before they can be generated. You can enable and disable events by specifying the appropriate values for queue manager or queue attributes (or both) depending on the type of event. You do this using:

- PCF commands
- MQSC (MQSeries) commands
- MQMT Master Terminal transactions

Enabling queue manager events

Queue manager events can be enabled or disabled by changing queue manage attributes. This is possible using the master terminal transaction, MQMT option 1.1, PF11. This option displays the queue manager's event settings, where each event type can be enabled or disabled.

Alternatively, the queue manager attributes can be set using PCF (see Chapter 8) or MQSeries commands (see Chapter 9).

Enabling channel events

Channel events are enabled via the queue manager event settings, however, channel events can be suppressed by not defining the channel events queue, or by making it put-inhibited. Note that this could cause a queue to fill up if remote event queues point to a put-inhibited channel events queue.

If a queue manager does not have a SYSTEM.ADMIN.CHANNEL.EVENT queue, or if this queue is put inhibited, all channel event messages are discarded unless they are being put by an MCA across a link to a remote queue. In this case they are put on the dead-letter queue.

Enabling performance events

Performance events as a whole must be enabled on the queue manager, otherwise no performance events can occur. You can then enable the specific performance events by setting the appropriate queue attribute. You also have to specify the conditions that give rise to the event.

Queue specific event settings are available via the local queue extended definition, available via MQMT option 1.2.

```
07/22/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
14:11:26                Queue Extended Definition                    CIC1
MQMMQUE                                                              A000

Object Name: ANYQ

General               Maximums                   Events
Type   . . : Local    Max. Q depth . : 00001000  Service int. event: N
File name  : MQFI001  Max. msg length: 00004096  Service interval  : 00000000
Usage  . . : T        Max. Q users . : 00000100  Max. depth event  : N
Shareable  : Y        Max. gbl locks : 00000200  High depth event  : N
                      Max. lcl locks : 00000200  High depth limit  : 000
                                                 Low depth event . : N
Triggering                                       Low depth limit . : 000
Enabled  . : N        Transaction id.:
Type . . . : E        Program id . . :
Max. starts: 0001     Terminal id  . :
Restart  . : N        Channel name . :
User data  :
           :
Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
            PF9=List  PF10=Queue
```

Figure 53. Extended definition

Enabling queue depth events

By default, all queue depth events are disabled. To configure a queue for any of the queue depth events you must:

Enable performance events on the queue manager
Enable the event on the required queue
Set the limits, if required, to the appropriate levels, expressed as a percentage of the maximum queue depth

Enabling queue service interval events

To configure a queue for queue service interval events you must:

Enable performance events on the queue manager, for example, using the queue manager attribute PerformanceEvent (PERFMEV in MQSC).
Set the control attribute, QServiceIntervalEvent, for a Queue Service Interval High or OK event on the queue, as required (QSVCIEV in MQSC).
Specify the service interval time by setting the QServiceInterval attribute for the queue to the appropriate length of time (QSVCINT in MQSC).

For example, to enable Queue Service Interval High events with a service interval time of 10 seconds (10 000 milliseconds) use the following MQSC commands:

```
ALTER QMGR   +
      PERFMEV(ENABLED)
ALTER QLOCAL('MYQUEUE') +
      QSVCINT(10000) +
      QSVCIEV(HIGH)
```

Note: When enabled, a queue service interval event can only be generated on an MQPUT call or an MQGET call. The event is not generated when the elapsed time becomes equal to the service interval time.

Automatic enabling of queue service interval events

The high and OK events are mutually exclusive; that is, when one is enabled, the other is automatically disabled.

When a high event is generated on a queue, the queue manager automatically disables high events and enables OK events for that queue.

Similarly, when an OK event is generated on a queue, the queue manager automatically disables OK events and enables high events for that queue.

Event queues

You can define event queues either as local queues, alias queues, or as local definitions of remote queues. If you define all your event queues as local definitions of the same remote queue on one queue manager, you can centralize your monitoring activities.

You must not define event queues as transmission queues because event messages have formats that are incompatible with the format of messages required for transmission queues.

Note: Attributes related to events for queues can be modified using the MQSET MQI call, commands, or the master terminal transactions.

If an event occurs when the event queue is not available, the event message is lost. For example, if you do not define an event queue for a category of event, all event messages for that category will be lost. The event messages are not, for example, saved on the dead-letter (undelivered-message) queue.

However, the event queue may be defined as a remote queue. Then, if there is a problem on the remote system putting messages to the resolved queue the event message will appear on the remote system's dead-letter queue.

An event queue might be unavailable for many different reasons including:

- The queue has not been defined.
- The queue has been deleted.
- The queue is full.
- The queue has been put-inhibited.

The absence of an event queue does not prevent the event from occurring. For example, after a performance event, the queue manager changes the queue attributes and resets the queue statistics. This happens whether the event message is put on the performance event queue or not.

You can set up the event queues with triggers so that when an event is generated, the event message being put onto the event queue starts a (user-written) monitoring application. This application can process the event messages and take appropriate action. For example, certain events may require that an operator be informed, other events may start off an application that performs some administration tasks automatically.

Format of event messages

Event messages contain information about the event and its origin. Typically, these messages are processed by a system management application program tailored to meet the requirements of the enterprise at which it runs. As with all MQSeries messages, an event message has two parts:

- a message descriptor
- the message data

The message descriptor is based on the MQMD structure, which is defined in the MQSeries Application Programming Reference manual. The message data is also made up of an event header and the event data. The event header contains the reason code that identifies the event type. The putting of the event message and any subsequent actions arising do not affect the reason code returned by the MQI call that caused the event. The event data provides further information about the event.

Message descriptor (MQMD) in event messages

The format of the message descriptor is defined by the MQSeries MQMD dat structure, which is found in all MQSeries messages and is described in the MQSeries Application Programming Reference manual. The message descriptor contains information that can be used by a user-written system monitoring application. For example:

- The message type
- The format type
- The date and time that the message was put on the event queue

In particular, the information in the descriptor informs a system management application that the message type is MQMT_DATAGRAM, and the message format is MQFMT_EVENT.

In an event message, many of these fields contain fixed data, which is supplied by the queue manager that generated the message. The MQMD also specifies the name of the queue manager (truncated to 28 characters) that put the message, and the date and time when the event message was put on the event queue.

Message data in event messages

The event message data is based on the programmable command format (PCF) which is used in PCF command inquiries and responses. The event message consists of two parts: the event header and the event data.

Event header

The MQCFH structure is the header structure used for event messages and for PCF messages. When the structure is used for event messages, the message descriptor Format field is MQFMT_EVENT. The datatype of the following parameters (MQLONG) is described in the MQSeries Application Programming Reference manual.

Event data

Event data varies depending on the event message. Event data uses the PCF data structures MQCFIN and MQCFST.

Event messages

MQSeries for VSE/ESA supports the following event messages:

- Alias Base Queue Type Error
- Channel Conversion Error
- Channel Started

- Channel Stopped
- Channel Stopped By User
- Get Inhibited
- Not Authorized (type 1)
- Not Authorized (type 2)
- Not Authorized (type 4)
- Put Inhibited
- Queue Depth High
- Queue Depth Low
- Queue Full
- Queue Manager Active
- Queue Manager Not Active
- Queue Service Interval High
- Queue Service Interval OK
- Remote Queue Name Error
- Transmission Queue Type Error
- Transmission Queue Usage Error
- Unknown Alias Base Queue
- Unknown Object Name
- Unknown Remote Queue Manager
- Unknown Transmission Queue

Viewing error logs

MQSeries error messages, and other system messages, are placed on the following queues:

SYSTEM.LOG
All MQSeries generated error messages are written to this queue. If SYSTEM.LOG is not defined, or if MQSeries cannot successfully write to it, the error messages are logged to CSMT and may be viewed using standard system utilities. The CSMT redirection parameter is an active toggle, and can be set in the global system definition.

SYSTEM.DEAD.LETTER.QUEUE
Is the MQSeries dead letter queue. Messages that cannot be queued to their specified destination are queued here.

SYSTEM.MONITOR
API monitor queue used to log all application requests and their results. This is primarily for problem determination purposes.

Notes:

1. The names listed for these queues are the default names, but you can redefine the actual queue names through the global system definition screen.
2. You can view the messages written to these queues using the MQSeries browse queue function (see “Browse function” on page 113).
3. The messages included in the SYSTEM.LOG can be controlled using the ’Log and Trace Settings’ screen. Refer to “Queue Manager Log and Trace Settings” on page 72.

Chapter 5. Utilities and interfaces

MQSeries for VSE/ESA is supplied with various utility functions, which are:

- The MQSeries System Administration Control Interface – see "System Administration Control Interface"
- Batch modules – see "Background batch modules" on page 135
- The batch interface – see "Using the batch interface" on page 136
- Utilities for reclaiming VSAM file space – see "VSAM file maintenance" on page 140

System Administration Control Interface

The MQSeries System Administration Control Interface allows a limited number of system administration functions to be performed using programs.

The System Administration Control Interface has:

- A transactional interface (MQCL)
- A programmable interface (MQPCMD)

Transactional interface (MQCL)

MQCL is the command-line interface to the MQPCMD program. It allows you selectively to:

- Stop, start or query Inbound and Outbound queues
- Open, close or query Sender, Receiver and Client channels

The command may be entered on cleared CICS terminal or on the VSE/ESA console as a command to the CICS partition.

The format of the command is:

```
MQCL FF NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

Where:

MQCL
Command line CICS transaction ID.

FF Function code. Valid codes are:

OS	Open Sender channel
CS	Close Sender channel
QS	Query Sender channel
OR	Open Receiver channel
CR	Close Receiver channel
QR	Query Receiver channel
OC	Open Client channel
CC	Close Client channel
QC	Query Client channel
SI	Start queue Inbound direction
XI	Stop queue Inbound direction
QI	Query queue Inbound direction
SO	Start queue Outbound direction
XO	Stop queue Outbound direction
QO	Query queue Outbound direction
SB	Start queue Both directions

© Copyright IBM Corp. 1995, 2004

XB Stop queue Both directions
QB Query queue Both directions
QD Query queue Depth

NNNN
The name of a queue or channel.

A message is sent to the activated terminal for every activation of this transaction. The original command is redisplayed if entered in a CICS session, eg

```
MQCL QB IBM.REPLY.QUEUE
   MQM001000 QUEUE STATUS: IN=IDLE    ,OUT=IDLE    IBM.REPLY.QUEUE
```

This message has a message code of MQM001000 for completed messages, or an error code of MQM001090 for any that did not complete properly. The text that follows explains the exact results.

Note that error code MQM001090 indicates invalid syntax in the command line program, MQCL. If an invalid function is entered then following help lines will be displayed:

```
MQCL XX IBM.REPLY.QUEUE
        MQCL is a command line interface which allows queues and
        channels to be selectively opened and closed. It has the
        flexibility to open and close in inbound, outbound or both
        directions. The syntax format is:

        MQCL FF NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

        where NN...NN = Queue or Channel name
              FF = one of following function codes
        CHANNEL - Type       Open   Close  Query
                  SENDER      OS     CS     QS
                  RECEIVER    OR     CR     QR
                  CLIENT      OC     CC     QC

        QUEUE   - Direction  Start  Stop   Query
                  INBOUND     SI     XI     QI
                  OUTBOUND    SO     XO     QO
                  BOTH        SB     XB     QB
        QUEUE Depth                       QD
        MQM001090 Command Line Program invalid syntax.
```

Programmable interface (MQPCMD)

The programmable interface uses an EXEC CICS LINK PROGRAM (MQPCMD). MQPCMD accepts a fixed format COMMAREA which specifies the type of request and a status response area.

The supplied copybook MQICMD.C describes this area.

```
*-----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQICMD    ***          - BEGIN - *
*-----------------------------------------------------------------*
*  COMMAND LINE COPYBOOK                                          *
*-----------------------------------------------------------------*

 01  MQI-COMMAND-LINE.
     02  MQI-CMD-PASSED-AREA.
        05  MQI-CMD-TRANS-ID         PIC  X(4) VALUE 'MQCL'.
        05  FILLER                   PIC  X    VALUE SPACE.
        05  MQI-CMD-FUNCTION         PIC  XX   VALUE SPACE.
             88 MQI-CMD-FUNCTION-OK    VALUE 'CR', 'OR', 'QR',
                                             'CS', 'OS', 'QS',
                                             'CC', 'OC', 'QC',
                                             'QB', 'QI', 'QO',
```

```
                                        'QD',
                                        'XB', 'XI', 'XO',
                                        'SB', 'SI', 'SO'.
           88 MQI-CMD-FUNC-CHANNEL   VALUE 'CR', 'OR', 'QR',
                                        'CS', 'OS', 'QS',
                                        'CC', 'OC', 'QC'.
           88 MQI-CMD-CLOSE-CHANNEL  VALUE 'CR', 'CS', 'CC'.
           88 MQI-CMD-OPEN-CHANNEL   VALUE 'OR', 'OS', 'OC'.
           88 MQI-CMD-QUERY-CHANNEL  VALUE 'QR', 'QS', 'QC'.
           88 MQI-CMD-CHANNEL-SEND   VALUE 'CS', 'OS', 'QS'.
           88 MQI-CMD-CHANNEL-RECVR  VALUE 'CR', 'OR', 'QR'.
           88 MQI-CMD-CHANNEL-CLIENT VALUE 'CC', 'OC', 'QC'.

           88 MQI-CMD-FUNC-QUEUE       VALUE 'XB', 'XI', 'XO',
                                            'SB', 'SI', 'SO',
                                            'QB', 'QI', 'QO',
                                            'QD'.
           88 MQI-CMD-STOP-QUEUE       VALUE 'XB', 'XI', 'XO'.
           88 MQI-CMD-STOP-Q-INBOUND   VALUE 'XI'.
           88 MQI-CMD-STOP-Q-OUTBOUND  VALUE 'XO'.
           88 MQI-CMD-STOP-Q-BOTH      VALUE 'XB'.

           88 MQI-CMD-START-QUEUE      VALUE 'SB', 'SI', 'SO'.
           88 MQI-CMD-START-Q-INBOUND  VALUE 'SI'.
           88 MQI-CMD-START-Q-OUTBOUND VALUE 'SO'.
           88 MQI-CMD-START-Q-BOTH     VALUE 'SB'.

           88 MQI-CMD-QUERY            VALUE 'QB', 'QI', 'QO',
                                            'QS', 'QR', 'QC',
                                            'QD'.
           88 MQI-CMD-QUERY-QUEUE      VALUE 'QB', 'QI', 'QO',
                                            'QD'.
           88 MQI-CMD-QUERY-Q-INBOUND  VALUE 'QI'.
           88 MQI-CMD-QUERY-Q-OUTBOUND VALUE 'QO'.
           88 MQI-CMD-QUERY-Q-BOTH     VALUE 'QB'.
           88 MQI-CMD-QUERY-Q-DEPTH    VALUE 'QD'.

           88 MQI-CMD-Q-IN             VALUE 'SI', 'XI'.
           88 MQI-CMD-Q-OUT            VALUE 'SO', 'XO'.
           88 MQI-CMD-Q-BOTH           VALUE 'SB', 'XB'.

        05  FILLER                  PIC  X   VALUE SPACE.
        05  MQI-CMD-QUEUE-NAME      PIC  X(48) VALUE SPACES.
        05  MQI-CMD-CHANNEL-NAME    REDEFINES
            MQI-CMD-QUEUE-NAME     PIC  X(48).

*--------values returned when LINKed
     02  MQI-CMD-RETURNED-AREA.
        05  MQI-CMD-RC              PIC S9(4) COMP VALUE ZERO.
         88 MQI-CMD-RC-OK                        VALUE ZERO.
         88 MQI-CMD-RC-DUPLICATE-FUNC            VALUE +4.
         88 MQI-CMD-RC-NAME-INVALID              VALUE +10.
         88 MQI-CMD-RC-SYS-NOT-ACTIVE            VALUE +80.
         88 MQI-CMD-RC-ERRORS                    VALUE +90.
         88 MQI-CMD-RC-HELP                      VALUE +99.

        05 MQI-CMD-ERROR-LINE.
          10  MQI-CMD-ERROR-PREFIX   PIC  XXX   VALUE 'MQM'.
          10  MQI-CMD-ERROR-CODE     PIC  X(6)  VALUE SPACES.
          10  FILLER                 PIC  X     VALUE SPACE.
          10  MQI-CMD-ERROR-TEXT     PIC  X(40) VALUE SPACES.
          10  FILLER                 PIC  X     VALUE SPACE.
          10  MQI-CMD-ERROR-NAME     PIC  X(48) VALUE SPACES.

        05  MQI-CMD-QUERY-STATES.
          10  MQI-CMD-QUERY-RC-INBOUND    PIC X(8) VALUE SPACES.
              88 MQI-CMD-QUERY-RC-IN-MAX      VALUE 'MAX     '.
```

```
                88 MQI-CMD-QUERY-RC-IN-FULL     VALUE 'FULL    '.
                88 MQI-CMD-QUERY-RC-IN-ERRORED  VALUE 'ERRORED '.
                88 MQI-CMD-QUERY-RC-IN-IDLE     VALUE 'IDLE    '.
                88 MQI-CMD-QUERY-RC-IN-ACTIVE   VALUE 'ACTIVE  '.
                88 MQI-CMD-QUERY-RC-IN-INHIBIT  VALUE 'INHIBIT '.
                88 MQI-CMD-QUERY-RC-IN-RECOVER  VALUE 'RECOVER '.
                88 MQI-CMD-QUERY-RC-IN-STOPPED  VALUE 'STOPPED '.
             10  MQI-CMD-QUERY-RC-CHANNEL
                 REDEFINES  MQI-CMD-QUERY-RC-INBOUND  PIC X(8).
                88 MQI-CMD-QUERY-RC-ABENDED     VALUE 'ABENDED '.
                88 MQI-CMD-QUERY-RC-BUSY        VALUE 'BUSY    '.
                88 MQI-CMD-QUERY-RC-CLOSED      VALUE 'CLOSED  '.
                88 MQI-CMD-QUERY-RC-IDLE        VALUE 'IDLE    '.
                88 MQI-CMD-QUERY-RC-UNKNOWN     VALUE 'UNKNOWN '.
             10  FILLER REDEFINES
                 MQI-CMD-QUERY-RC-INBOUND.
                 15 MQI-CMD-QUERY-RC-DEPTH   PIC S9(8) COMP.
                 15 FILLER                   PIC X(4).
             10  MQI-CMD-QUERY-RC-OUTBOUND   PIC X(8) VALUE SPACES.
                88 MQI-CMD-QUERY-RC-OUT-ERRORED VALUE 'ERRORED '.
                88 MQI-CMD-QUERY-RC-OUT-IDLE    VALUE 'IDLE    '.
                88 MQI-CMD-QUERY-RC-OUT-ACTIVE  VALUE 'ACTIVE  '.
                88 MQI-CMD-QUERY-RC-OUT-INHIBIT VALUE 'INHIBIT '.
                88 MQI-CMD-QUERY-RC-OUT-RECOVER VALUE 'RECOVER '.
                88 MQI-CMD-QUERY-RC-OUT-STOPPED VALUE 'STOPPED '.

*-----------------------------------------------------------------*
* - END -        ***  COPYBOOK: MQICMD    ***            - END - *
*-----------------------------------------------------------------*
```

The following sample program is an example of how to use MQICMD.C in a CICS application program:

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. TESTCMD.
       AUTHOR. IBM.
       DATE-COMPILED.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  WS-QCLOSED  PIC X(20) VALUE 'Queue closed OK.'.
       01  WS-QNCLOSED PIC X(20) VALUE 'Queue not closed.'.

           COPY MQICMD.

       PROCEDURE DIVISION.

       0000-MAIN-LINE.
           MOVE 'ANYQ'                TO MQI-CMD-QUEUE-NAME.
           SET  MQI-CMD-STOP-Q-BOTH   TO TRUE.
           EXEC CICS LINK
                        PROGRAM('MQPCMD')
                        COMMAREA(MQI-COMMAND-LINE)
                        LENGTH(LENGTH OF MQI-COMMAND-LINE)
           END-EXEC.
           EVALUATE TRUE
              WHEN MQI-CMD-RC-OK
                 EXEC CICS WRITE OPERATOR
                                 TEXT(WS-QCLOSED)
                                 TEXTLENGTH(LENGTH OF WS-QCLOSED)
                 END-EXEC
              WHEN OTHER
                 EXEC CICS WRITE OPERATOR
                                 TEXT(WS-QNCLOSED)
                                 TEXTLENGTH(LENGTH OF WS-QNCLOSED)
```

```
            END-EXEC
        END-EVALUATE.

    0000-RETURN.
        EXEC CICS RETURN
        END-EXEC.

        GOBACK.
```

Background batch modules

The PRD2.MQSERIES library contains all the sample code and JCL, including the MQJUTILY.Z example background batch job.

MQJUTILY.Z contains the MQPUTIL program, which performs the following functions:

- Prints the system, queue, and channel definitions from a configuration file.
- Prints the SYSTEM.LOG file in a formatted report.
- Updates all channels with a new starting MSN.
- Updates a configuration file for dual queues. It makes all dual queues into a primary queue.
- Prints new Help Facility error information.

The MQPUTIL program uses the CONFIG DLBL for the MQSeries configuration VSAM file, if the PRINT LOG command is used. The MQPUTIL program uses the following general syntax:

Table 8. MQPUTIL program general syntax

Column	Content
1 to 5	Command name
6	Space
7 to 18	Subcommand
19	Space
20...	Arguments

MQPUTIL program

The supplied MQPUTIL program has three commands, which are:

- PRINT
- RESET
- DUALQ

Note: You must run the MQPUTIL program once for each command that you require.

PRINT

PRINT has three options:

CONFIG
Prints the full MQSeries configuration using the sample DLBL provided.

LOG Prints the system log in a formatted report using the sample DLBL provided.

MESSAGES
Prints a Help Facility resolution report.

RESET

MSN nnnnnnnn
Resets all channel numbers to nnnnnnnn and checkpoint values to zero.

DUALQ

DUALQ has the following option:

TAKEOVER dual_queue_name
Allows the dual queue specified to become the primary queue, using the following process:

1. The configuration file points to the cluster hosting the dual queue instead of to the cluster hosting the primary queue.
2. All message headers in the dual queue are modified to contain the name of the primary queue instead of the name of the dual queue.

This command may be used when a local queue becomes unavailable, for example, when input or output errors occur, and a dual queue has been defined.

Note: You are recommended to backup the configuration file, using the VSAM REPRO command, before using this command, because the file will be changed. The configuration file can be restored when you have repaired the failure.

Refer to the sample JCL stream in Appendix D, “Sample JCL and programs,” on page 365.

Using the batch interface

Unlike MQSeries for other platforms, MQSeries for VSE/ESA is implemented as a CICS subsystem. This means that access to MQSeries objects using the message queue interface (MQI) is restricted to CICS applications. To avoid this limitation, MQSeries for VSE/ESA provides an interface for batch programs.

The batch interface is designed to standardize the programming style of CICS and batch programs. From a programming point of view, batch programs use calls exactly the same way as CICS programs, that is, MQSeries batch programs issue calls such as MQCONN, MQOPEN, and MQPUT to access MQSeries objects.

Because MQSeries objects are ultimately under the control of the CICS subsystem, calls issued by batch programs are passed to the CICS partition for processing. This is achieved using cross partition communication calls (XPCC). Batch programs are not concerned with XPCC, because all relevant logic is built into MQI calls.

MQSeries for VSE/ESA provides a special CICS transaction, MQBI, that must be running to process MQI calls issued by batch programs. This transaction must be running for the batch interface to be available. MQBI waits for MQCONN calls issued by batch programs. When these are received, MQBI starts a second transaction, MQBX, which issues all MQI calls on behalf of the batch program. There is one MQBX instance for each active batch connection.

The MQBX transaction runs for the duration of the logical MQSeries connection, that is, it runs until the batch program issues an MQDISC. If a batch program issues a second MQCONN call, the batch interface starts a second MQBX transaction for the duration of that MQSeries connection. This design allows batch programs to create logical units of work. It also means that multiple batch programs (including multiple VSE subtasks) can establish concurrent connections to the MQSeries queue manager.

Note: Using the batch interface adds a performance overhead, because MQI calls issued from batch programs are transferred to mirror CICS transactions.

Each MQSeries queue manager running on a VSE/ESA system can activate a single batch interface. Each interface is identified by a unique batch identifier name. Batch identifiers must be unique within the context of the VSE/ESA system (that is, remote VSE/ESA systems can use the same identifier names, but queue managers running on the same VSE/ESA system must be configured with unique identifiers).

The queue manager's batch identifier is configured in the global system definition as a communications parameter (press PF9 from MQMT option 1.1):

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
14:47:42                   Global System Definition                  CIC1
MQMMSYS                     Communications Settings                  A000

    TCP/IP settings                          Batch Interface settings
    TCP/IP listener port : 01414             Batch Int. identifier: MQBISERV
    Licensed clients . . : 00000             Batch Int. auto-start: Y
    Adopt MCA  . . . . . : Y
    Adopt MCA Check  . . : Y

    SSL parameters
    Key-ring sublibrary  : PRD2.SSLKEYS
    Key-ring member  . . : MQVSEKEY

    PCF parameters
    System command queue : SYSTEM.ADMIN.COMMAND.QUEUE
    System reply queue . : SYSTEM.ADMIN.REPLY.QUEUE
    Cmd Server auto-start: Y
    Cmd Server convert . : N
    Cmd Server DLQ store : Y

Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 54. Batch interface identifier

The batch interface settings, as part of the communication parameters of the queue manager's global system definition, include the batch interface identifier and the batch interface auto-start parameters.

Batch interface identifier

The batch interface identifier is an 8-character alpha-numeric name that uniquely identifies the queue manager to batch MQI programs.

Batch programs identify which queue manager to connect to by including a `// SETPARM` card in their JCL. The SETPARM parameter required to identify a queue manager is MQBISRV. For example:

```
// SETPARM MQBISRV='MQBISRV2'
```

Since batch JCL can only specify one SETPARM for the MQBISRV symbol, batch programs can only connect to one queue manager per submission, even if there are multiple queue managers running on the VSE/ESA system.

The default value for the batch interface identifier is MQBISERV. This is also the default for batch programs that do not supply a //SETPARM card in their JCL.

If two queue manager's running on the same VSE/ESA system rely on the identifier default, or are configured with the same identifier name, the queue manager that services batch programs is unpredictable.

Batch interface auto-start

The batch interface auto-start parameter indicates whether or not the queue manager should automatically start the batch interface (transaction MQBI) during system initialization.

The auto-start parameter can be set as follows:

Y Start the batch interface during queue manager initialization.

N Do not start the batch interface during queue manager initialization.

By choosing to start the batch interface during queue manager initialization, the batch interface will also be stopped automatically during queue manager shutdown.

Starting the batch interface

The batch interface is started by running the MQBI transaction. This can be done in native CICS, or by configuring CICS to run the batch interface start program (MQPSTBI) during post initialization. MQBI is a long-running CICS transaction that coordinates multiple simultaneous batch connections to the MQSeries queue manager.

Alternatively, the batch interface can be started automatically during queue manager initialization by switching on the batch interface auto-start parameter in the communications settings of the global system definition.

Stopping the batch interface

The batch interface can be stopped normally in one of three ways:

- in native CICS
- from a batch program
- during CICS system shutdown

The batch interface can be stopped abnormally by shutting down CICS with the immediate option, or by purging or forcing the MQBI transaction. The MQBISTOP sample program provides an example of stopping the interface from a batch program.

The batch interface can be stopped during CICS system shutdown by configuring the CICS PLTSD to run program MQPSPBI. If so configured, the PLT macro for MQPSPBI should precede the MQ shutdown program MQPSTOP. For example:

```
DFHPLT TYPE=ENTRY,PROGRAM=MQPSPBI
DFHPLT TYPE=ENTRY,PROGRAM=MQPSTOP
```

The batch interface is stopped automatically if the batch interface auto-start parameter is set on in the communications settings of the global system definition.

How to use the batch interface

1. Issue MQSeries functions in your batch program, just as you do in CICS programs. For example:

```
CALL 'MQCONN'  USING
     QM-NAME-AREA
     HCONN-ADDR-AREA
     CCODE-ADDR-AREA
     RCODE-ADDR-AREA.
```

2. Link-edit your program by including module MQBIBTCH. For example:

```
// JOB MQBATBLD
// OPTION CATAL
PHASE MYPROG,*
// EXEC IGYCRCTL,....
    your program here
/*
INCLUDE MQBIBTCH
// EXEC LNKEDT
/&
```

3. Start the CICS interface, using the transaction MQBI.
4. Run your batch program.

   The following JCL might be used to run a MQSeries batch application called MYPROG:

```
// JOB MQBATRUN
// SETPARM MQBISRV='MQBISRV2'
// LIBDEF *,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
// EXEC MYPROG
/*
/&
```

   Note that the SETPARM card identifies the queue manager the batch program will connect to, and the LIBDEF SEARCH path must include the MQSeries installation library. If you omit the SETPARM for the MQBISRV symbol, the batch program will attempt to communicate with a queue manager configured with the batch identifier name MQBISERV.

Data integrity

To test for data integrity the following functions are used:

- MQCMIT commits all changes. This forces a CICS SYNCPOINT to be issued by the mirror transaction.
- MQBACK rolls back all changes. The CICS mirror transaction issues `EXEC CICS SYNCPOINT ROLLBACK`.

For both functions the syntax is as follows:

```
CALL 'funct' USING
    HCONN-ADDR-AREA
    CCODE-ADDR-AREA
    RCODE-ADDR-AREA.
```

Notes:

1. None of the passed parameters is actually tested or used.
2. Under CICS, updates are not automatically committed. However, if a batch program issues the MQDISC call while there are uncommitted requests, an implicit syncpoint occurs.

Verifying the batch interface

The batch program MQBICALL has been provided for this purpose. You can use the following job as a test:

```
// JOB CALLER
// SETPARM MQBISRV='batchid'
// LIBDEF *,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
```

```
*  Put 5 messages into queue: ANYQ
// EXEC MQBICALL
PUT  05 ANYQ
/*
/&
```

The MQBICALL utility is provided with the MQSeries installation library as both an executable and a COBOL source file. It can be used as a basis for your own MQSeries batch applications.

Note that the SETPARM for symbol MQBISRV should be changed to identify an appropriate queue manager batch interface identifier.

Restrictions on using the batch interface

1. Message lengths are restricted to 250k.
2. The MQINQ and MQSET MQI calls are limited as follows:
   - A maximum of 10 selectors.
   - A maximum of 10 integer attributes.
   - 500 characters for character attribute buffer.

VSAM file maintenance

All files used by MQSeries are VSAM clusters. Most of these contain queues and need to be reorganized from time to time.

A queue is an ordered suite of VSAM records in a KSDS organization. Each record key is 56 bytes long, 48 being for the queue name, and eight for the Queue Sequence Number (QSN) and other information. This QSN is assigned sequentially, resulting in all keys being created in ascending order.

Even when a queue record is physically deleted from a queue, the space it occupies is not reclaimed due to the way VSAM works. Therefore, unless you reclaim the space used by these records, there is the possibility that you will obtain a VSAM “space full” condition.

The queue dump facility allows you to rebuild an MQSeries VSAM queue file. This eliminates processed messages and fully regains VSAM freespace.

There are three ways to reclaim the space of deleted messages :

1. Use the MQPREORG utility.
2. Perform a VSAM DELETE and DEFINE to recreate the VSAM dataset. Only do this if your queue is empty. If you have multiple queues in a single VSAM file (not recommended), all queues should be empty.
3. Use the automatic reorganization feature available with your queue definition. Automatic reorganization is available only for single queues defined in a single VSAM file.

Delete all function

On the Maintain Queue Records screen (see “Queue maintenance” on page 107), there is a function called “Delete All”. This function physically deletes all messages, and resets the QSN to one, in order to reclaim freed space.

This is a useful tool to maintain the system log file for MQSeries. The advantage of this function is that it is an online function requiring no other manual operation.

Attention: Note that this function deletes all messages and should not be used on queue files that contain undelivered messages. It is not recommended that the DELETE ALL option be used to maintain production queues as this can leave VSAM indexes fragmented and lead to poor performance. Instead, the automatic reorganization feature or the batch MQPREORG utility should be considered.

Operation

1. On the Start/Stop Queue Control screen, stop the desired queue; see Figure 41 on page 102.
2. If the desired queue is a transmission queue, stop only the inbound direction first. When the queue depth reaches zero, stop the outbound direction and close the associated sender channel.
3. If the desired queue is a destination queue with trigger capability, close the associated receiver channel.
4. On the Maintain Queue Records screen enter the queue name, together with a function of A, and press the PF6 (Update) function key; see Figure 45 on page 108.
5. Press the Enter key to display the result.
6. After "Queue Processing Finished" is displayed, start the reorganized queue on the Start/Stop Queue control screen.

MQPREORG function

MQSeries includes a batch program utility called MQPREORG, and sample JCL to run MQPREORG.

This utility can be used as a nightly, or weekly, queue maintenance facility on any number of queue files. You can also specify a date and time to carry out the procedure. The utility accepts the queue name from SYSIPT and the name of the VSAM file from DLBL.

All messages are ignored, except those marked as "Written" (to be delivered after a specified date and time) on the specified queue. The retained messages are resequenced and placed in a work file.

After the VSAM cluster is deleted and redefined, the retained and resequenced messages are copied back into it. If none of the written messages is to be retained, you can use a "delete-and-define" IDCAMS JCL to do the job.

Multiple queues sharing a VSAM cluster

Attention: Although it is possible for MQSeries for VSE/ESA queues to share the same VSAM file cluster, this is **not** advised. To give maximum independence to data, each queue should be assigned a unique VSAM file cluster.

This is particularly important if the queue is defined for automatic reorganization. See "Creating local queues" on page 77.

If there is more than one queue defined in a VSAM cluster, all queues have to be processed before deleting and recreating this cluster. Otherwise, records from unprocessed queues will be lost.

To help you reorganize all queues, you may use the "ALL" option instead of the queue name, as follows:

```
// EXEC MQPREORG
ALL
/*
```

To reorganize a specific queue, enter one of the following commands:

```
// EXEC MQPREORG
LQ.INVOICE
/*
```

or

```
// EXEC MQPREORG
LQ.INVOICE YYYYNNDDHHMMSS
/*
```

where:

YYYY	Is the year
NN	Is the month
DD	Is the day
HH	Is the hour
MM	Is the minute
SS	Is the second

Reorganizing queue files

This procedure is to be used only when the queue manager is not running.

1. If CICS is running, use CEMT to shut down and close the VSAM files you are going to process.
2. Modify the sample JCL to include your system parameters and reorganization requirements.
3. Process the job to run the batch program utility, MQPREORG, to reorganize the VSAM files and reclaim all freed space.
4. If you performed step 1, use CEMT to open and enable the processed VSAM files.

Sample JCL to run MQPREORG

```
* ** JOB JNM=MQJREORG,DISP=D,CLASS=0
* ** LST DISP=H,CLASS=Q,PRI=3
// JOB MQJREORG - Re-Organize MQ/Series for VSE/ESA queues.
* ------------------------------------------------------------------*
*      I M P O R T A N T   I M P O R T A N T  I M P O R T A N T    *
*                                                                   *
*   Please change :                                                 *
*                 "*  ** JOB" to  "* $$ JOB"                        *
*                 "*  ** LST" to  "* $$ LST"                        *
*                 "*  ** EOJ" to  "* $$ EOJ"                        *
*                                                                   *
*   Fields filed with ?volid? have also to be modified to suit the  *
*  user specifications.                                             *
*                                                                   *
* ------------------------------------------------------------------*
*                                                                   *
*   This job deletes delivered messages from an MQSeries queue in   *
*   order to reclaim the DASD freed space.                          *
*                                                                   *
*   INPUT to MQPREORG :                                             *
*   (only one statement is allowed, delimited by one or more spaces)*
*                                                                   *
*     1. Any QUEUE name delimited by one or more spaces             *
*        (In this JCL, only queue OS2_LOCAL is to be processed)     *
```

```
*        If there are any other queues in the same cluster,         *
*        they will be echoed into OUTPUTQ.                          *
*     2. If you want to process EVERY queue in a cluster,           *
*        please key in "ALL ".                                      *
*                                                                   *
*   This sample assumes we want to reorganize queues defined to the *
*   VSAM cluster MQIF002. Changes must be made for other clusters.  *
* ------------------------------------------------------------------*
* Licensed Materials - Property of IBM                              *
*                                                                   *
* 5686-A06                                                          *
* (C) Copyright IBM Corp. 1998, 2002                                *
*                                                                   *
* US Government Users Restricted Rights - Use, duplication or       *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. *
* ------------------------------------------------------------------*
// DLBL INPUTQ,'MQSERIES.MQFI002',,VSAM,CAT=MQMCAT
// DLBL OUTPUTQ,'MQSERIES.WORK.QUEUE',,VSAM,CAT=MQMCAT
// EXEC IDCAMS,SIZE=AUTO
 /*                                                                 */
 /*                  VERIFY VSAM FILE                               */
 /*                                                                 */
          VERIFY FILE(INPUTQ)
          IF MAXCC > 0 THEN CANCEL   /* This means Cluster in use  */

          DELETE   (MQSERIES.WORK.QUEUE)                     -
                    CL ERASE PURGE CAT(?CAT?)
          SET MAXCC = 0
          DEFINE CLUSTER                                     -
                   (NAME (MQSERIES.WORK.QUEUE)               -
                    CYLINDERS (10 10)                        -
                    VOLUMES (?volid?)                        -
                    NONINDEXED)                              -
                 DATA                                        -
                   (NAME (MQSERIES.WORK.QUEUE.DATA)          -
                    RECORDSIZE (57   32703)                  -
                    CISZ (8096))                             -
                    CAT (?CAT?)
/*
// IF $MRC GT 0 THEN
// GOTO WRAPUP
// LIBDEF PHASE,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
// EXEC MQPREORG,SIZE=AUTO
OS2_LOCAL
/*
// IF $MRC GT 0 THEN
// GOTO WRAPUP
// EXEC IDCAMS,SIZE=AUTO
          DELETE   (MQSERIES.MQFI002)                          -
                    CLUSTER NOERASE PURGE CATALOG (?CAT?)
     SET MAXCC = 0
 /*                                                               */
          DEF CLUSTER(NAME(MQSERIES.MQFI002)                     -
              FILE(MQFI002)                                      -
              VOL(?volid?)                                       -
              RECORDS (3000 100)                                 -
              RECORDSIZE (200 4089)                              -
              INDEXED                                            -
              KEYS(56 0)                                         -
              SHR(2))                                            -
              DATA (NAME (MQSERIES.MQFI002.DATA) CISZ(4096))     -
              INDEX (NAME (MQSERIES.MQFI002.INDEX) CISZ(1024))   -
              CATALOG(?CAT?)
          IF LASTCC > 0 THEN  CANCEL
  /*                                                             */
  /*         Execute REPRO only of the define  was OK.           */
  /*                                                             */
```

```
        REPRO INFILE(OUTPUTQ) OUTFILE(INPUTQ)
        IF LASTCC > 0 THEN  CANCEL
  /*                                                              */
  /*             Delete only if REPRO was OK.                     */
  /*                                                              */
        DELETE   (MQSERIES.WORK.QUEUE)                    -
                   CL ERASE PURGE CAT(?CAT?)
/*
/. WRAPUP
/&
* ** EOJ
```

Chapter 6. Problem determination

This chapter suggests reasons for some of the problems you may have using MQSeries for VSE/ESA. The process of problem determination is that you start with the symptoms and trace them back to their cause.

Not all problems can be solved immediately, for example, performance problems caused by the limitations of your hardware. Also, if you think that the cause of the problem is in the MQSeries code, contact your IBM Support Center.

The cause of your problem could be in:
- MQSeries setup and local queue operation
- The network
- The application
- Other areas of investigation

The sections that follow raise some fundamental questions that you need to consider. Work through the questions, making a note of anything that might be relevant to the problem.

MQSeries setup and local queue operation

You should ensure that MQSeries for VSE/ESA is installed correctly and working with local queues before you investigate any other problems.

Has MQSeries run successfully before?

If MQSeries has not run successfully before, it is likely that you have not yet set it up correctly. See "MQSeries installation verification test" on page 23 to check that you have carried out all the steps correctly, and set up a SYSTEM.LOG queue as follows:

1. Define a queue name as SYSTEM.LOG using:
   a. A physical file name MQFLOG using the file name from the file control table
   b. A maximum queue depth of 1 000 000

   See step 4d on page 21 and step 4f on page 21 in "MQSeries installation verification test" on page 23 for information about defining a queue.
2. Enter SYSTEM.LOG as the object name with a valid QSN number.

You can browse the log queue by selecting 4, Browse Queue Records, on the Master Terminal Main Menu, as described in "Browse function" on page 113.

See "Global system definition" on page 66 for more information.

Is local queue operation working?

This may require the creation of a local queue definition as described in "MQSeries initialization" on page 19. In addition, check that the VSAM files referenced in the queue definition are open and correctly enabled.

Use:

```
CEMT INQUIRE FILE (filename)
```

© Copyright IBM Corp. 1995, 2004

to ensure that the VSAM file associated with the queue is accessible.

Use the instructions described in "Local queue verification test" on page 23 to test a local queue. Ensure that you can:

- Put and get messages to the local queue, using the supplied test transaction TST2.
- Browse the queue correctly using the MQMT System Administration Browse function.

Network problems

Before MQSeries for VSE/ESA can use an inbound or outbound channel connection to an SNA-connected MQSeries platform, a connection must already be established between CICS/VSE and the remote platform.

The person responsible for the VTAM and CICS definitions in your enterprise should perform the following investigations.

Investigating SNA problems

If an attempt to start a channel fails, it may be the result of a session failure. If it is not possible to establish a session between CICS and the LU for the remote channel endpoint, either before or during the channel attempting to start, the connection fails.

Enter the following command if you suspect that a session failure is causing the problem:

```
D NET,ID=<remote lu name>,E
```

This gives details of the LU which should be in session with CICS, and also lists any sessions it currently has.

Notes:

1. Look at the session limit for the LU. If it is shown as one for an independent LU, there is a problem with the SNA definitions.
2. See if <minor node name> is listed amongst the sessions. If it is, there is a session between the LU and CICS. This indicates that the problem may not be at the network level, or that there are insufficient sessions between the two LUs to support a new channel request.

   Enter the command again, to see whether for this session, the send and receive counts have changed, indicating the session is in use.

If the command returns "PARAMETER VALUE INVALID", this means that VTAM does not know the <remote lu name>. Either you entered the name incorrectly, or VTAM cannot locate it. Try defining the name again and attempt to start the channel.

If VTAM is able to display <remote lu name>, try the following command in CICS:

```
CEMT I CONN(<remote conn>)
```

This shows the status of the connection from CICS to the remote system. Next to the entry is an indication showing it to be INService or OUTservice and ACQuired or RELeased. The status needs to be Inservice and Acquired.

```
CEMT I MODE CONN(<remote conn>)
```

This command displays the status of the mode names associated with the connection. For connections supporting parallel sessions, there will be at least two mode names, SNASVCMG and <logmode 1>, showing the number of active sessions for each.

If the SNASVCMG group has no sessions active, the connection is in a RELeased state, rather than an ACQuired state.

These sessions are SNA services manager sessions, and not used by MQSeries channels. However, at least one of the two needs to be active for the connection to be usable.

If the remote LU has been incorrectly defined, so that it has a session limit of one, it is possible that one SNASVSMG session is active, but that no other sessions can be established, including those required by the MQSeries channel.

The <logmode 1> sessions may be used by MQSeries channels.

For single session connections, one mode name, <logmode 2>, is shown with just one session in the group.

The MQSeries channel must have been set up to use the logon mode <logmode 1>, or <logmode 2>, as appropriate.

Investigating TCP/IP problems

Is TCP/IP able dynamically to establish a session between nodes in the network? Use the following instruction to test a connection to a remote TCP/IP node:

```
[ping hostname]
```

If you are unable to “ping” the remote TCP/IP node successfully, inform your VSE/ESA systems programmer who installed TCP/IP.

Under VSE/ESA, the ping command can be entered from the console. For example:

```
msg f7
AR 0015 1I40I  READY
F7-0111 IPN300I Enter TCP/IP Command
111 ping 127.0.0.1
F7 0109 TCP910I  Client manager connected. Generated on 10/28/01 at 23.57
F7 0109 TCP915I  PING
F7 0109 TCP910I  PING Ready:
F7 0109 TCP915I  SET HOST= 127.0.0.1
F7 0109 TCP910I  127.000.000.001
F7 0109 TCP910I  PING Ready:
F7 0109 TCP915I  PING
F7 0109 TCP910I  PING 1 was successful, milliseconds: 00011.
F7 0109 TCP910I  PING 2 was successful, milliseconds: 00000.
F7 0109 TCP910I  PING 3 was successful, milliseconds: 00000.
F7 0109 TCP910I  PING 4 was successful, milliseconds: 00000.
F7 0109 TCP910I  PING 5 was successful, milliseconds: 00003.
F7 0109 TCP910I  PING Ready:
F7 0109 TCP915I  QUIT
F7 0111 IPN300I Enter TCP/IP Command
```

Note that the IP address or hostname used should match the IP address or hostname used in a sender channel definition. For example, if messages are to be sent from VSE/ESA to a remote system, a sender channel will be defined that identifies that remote system by either IP address or hostname.

It should also be noted that some systems can deactivate or restrict ping requests. If this is the case, the TCP/IP administrator of the remote system should provide an alternative method to test connectivity between remote systems.

When messages are to be sent to VSE/ESA, the remote system should be able to ping the VSE/ESA system. Once again, if ping activity is restricted or disabled, the TCP/IP administrator should provide an alternative method to test connectivity between the two systems.

Investigating SSL problems

If secure sockets layer (SSL) services are configured for a sender, receiver or client channel, that channel may terminate prematurely if an SSL error occurs. Such failures generally involve error messages written to the SYSTEM.LOG. Assuming logging is active for error and critical messages, the SYSTEM.LOG should always be checked in the event of a channel failure. Note that the severity of messages logged to the system log can be set from MQMT option 1.1.

A channel is SSL enabled if the SSL Cipher Specification parameter is specified in the channel definition. All other SSL parameters are ignored if this parameter is not set (that is, it is left blank).

SSL enabled channels require SSL services installed and active on both the local and remote systems. In addition, the SSL configuration parameters associated with the local channel definition, generally, must match the parameters associated with the remote channel definition. These should be checked in the event of an SSL failure. Generally, SSL channels may fail due to one of the following situations:

- The key-ring sublibrary name defined in the Global System Definition communications parameters does not identify a valid SSL key-ring sublibrary.
- The key-ring member name defined in the Global System Definition communications parameters does not identify a valid member set in the key-ring sublibrary for .PRVK and .CERT files.
- The SSL cipher specification of the channel definition is invalid or not supported by both the local and remote SSL subsystems.
- SSL client authentication of the channel definition is required, but the remote system did not provide an X.509 PKI certificate during SSL initial negotiation.
- The SSL peer attributes of the channel definition identify specific features expected of the remote system's certificate that do not match.
- The local and remote SSL subsystems are incompatible or running different version levels.

Does the problem affect specific parts of the network?

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote message queue manager is not working, the messages cannot flow to a remote queue.

Check that the connection between the two systems is available, and that the intercommunication component of MQSeries has been started.

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

Have you made any network-related changes, or changed any MQSeries definitions, that might account for the problem?

Applications

The errors in the following list illustrate the most common causes of problems encountered while running MQSeries programs. You should consider the possibility that the problem with your MQSeries system could be caused by one or more of these errors:

- Assuming that queues can be shared, when they are in fact exclusive.
- Passing incorrect parameters in an MQI call.
- Passing insufficient parameters in an MQI call. This may mean that MQI cannot set up completion and reason codes for your application to process.
- Failing to check return codes from MQI requests.
- Passing variables with incorrect lengths specified.
- Passing parameters in the wrong order.
- Failing to initialize *MsgId* and *CorrelId* correctly.

Are there any error messages?

MQSeries uses the system log to capture messages concerning the operation of MQSeries itself, the queue manager, and error data coming from the channels that are in use. Check the system log to see if any messages have been recorded that are associated with your problem.

See "System log" on page 159 for information about the contents of the system log.

Are there any return codes explaining the problem?

If your application gets a return code indicating that a Message Queue Interface (MQI) call has failed, refer to the *MQSeries Application Programming Reference* manual for a description of that return code.

Can you reproduce the problem?

If you can reproduce the problem, consider the conditions under which it is reproduced:

- Is it caused by a command or an equivalent administration request?

  Does the operation work if it is entered by another method? If the command works if it is entered on the command line, but not otherwise, check that the command server has not stopped.
- Is it caused by a program? Does it fail on all MQSeries systems and all queue managers, or only on some?
- Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.

Have any changes been made since the last successful run?

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

- Have you changed, added, or deleted any queue definitions?
- Have you changed or added any channel definitions? Changes may have been made to either MQSeries channel definitions or any underlying communications definitions required by your application.

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

Has the application run successfully before?

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

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

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

  If so, it is likely that the error lies somewhere in the new or modified part of the application. Take a look at the changes and see if you can find an obvious reason for the problem. Is it possible to retry using a back level of the application?
- Have all the functions of the application been fully exercised before?

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

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

  Has your MQSeries system been changed, perhaps in a minor way, such that your application does not check the return codes it receives as a result of the change. For example, does your application assume that the queues it accesses can be shared? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?
- Does the application run on other MQSeries systems?

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

If the application has not run successfully before

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

Before you look at the code, and depending upon which programming language the code is written in, examine the output from the translator, or the compiler and linker, if applicable, to see if any errors have been reported.

If your application fails to translate, compile, or link, it will also fail to run if you attempt to invoke it.

If the documentation shows that each of these steps was accomplished without error, you should consider the coding logic of the application. Do the symptoms of the problem indicate the function that is failing and, therefore, the piece of code in error? See “Applications” on page 149 for some examples of common errors that cause problems with MQSeries applications.

Using the MQSeries API monitor

By selectively using the MQSeries API monitor, you can:
- Track precisely which MQSeries API issues an application
- Establish which return codes are passed

The MQSeries API monitor is started and stopped using the MQMT system administration transaction option 2.1, which is used to toggle the API monitor on and off.

```
12/31/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
12:03:36               Start / Stop Queue                           CIC1
MQWMSS                                                              A003

                          System Information
          System Status    : SYSTEM IS ACTIVE
          Queue Status     : Queuing System is active.
          Channel Status   : Channel System is active.
          Monitor Status   : Monitor is not active.
                         Single Queue Request
          Queue Name       :
          Function         :         S=Start, X=Stop,  R=Refresh from Config
          Mode             :         I=Inbound, O=Outbound, B=Both

          INBOUND Status   :
          OUTBOUND Status  :

                          Queuing System Request
          Function         : M       S=Start, X=Stop, or M=Monitor


Please enter a Queue name.
Enter=Display  PF2=Return  PF3=Exit  PF6=Update
```

Figure 55. API monitor

Once the API monitor is started, the application to be tested can be processed, and the API monitor stopped.

Note: The API monitor should be started for limited periods only. It traces the processing of all running applications, and consequently makes heavy usage of system resources.

After the API monitor is toggled off, the SYSTEM.MONITOR queue can be browsed using the MQMT system administration browse facility. Each message in the queue represents the result of an MQSeries API call.

```
08/07/2002          IBM MQSeries for VSE/ESA Version 2.1.2          IYBPZS01
11:16:43                  Browse Queue Records                      VSE1
MQMDISP                     SYSTEM IS ACTIVE                        A005

    Object Name: SYSTEM.MONITOR
    QSN Number : 00000006       LR-        0, LW-        8, DD-MQFO001
                     Queue Data Record
Record Status : Written.        PUT date/time  : 19980812110156
Message Size  : 00000428        GET date/time  :
Queue line.
TSTPA004....            01              OPEN    ....e.......e...........
     VSEP                                            QL.VSEP
                     QL.VSEP









Information displayed.
    5686-A06 (C) Copyright IBM Corp. 1998,2002  All Rights Reserved.
 ENTER = Process  PF2 = Main Menu    PF3 = Quit   PF4 = Next    PF5 = Prior
 PF7 = Up     PF8 = Down    PF9=Hex/Char   PF10=Txt/Head        PF12 = Monito
```

Figure 56. API monitor - browse

An MQOPEN request is displayed in Figure 56. However, for more information, display the text in hexadecimal format, as shown in Figure 57.

```
08/07/2002          IBM MQSeries for VSE/ESA Version 2.1.2          IYBPZS01
11:16:02                  Browse Queue Records                      VSE1
MQMDISP                     SYSTEM IS ACTIVE                        A005

    Object Name: SYSTEM.MONITOR
    QSN Number : 00000006       LR-        0, LW-        8, DD-MQFO001
                     Queue Data Record
Record Status : Written.        PUT date/time  : 19980812110156
Message Size  : 00000428        GET date/time  :
Queue line.
TSTPA004....            01              OPEN    ....e.......e...........  0000
EEEDCFFF000B444444444444FF44444444444444DDCD4444031981000319820000000000
32371004007800000000000001000000000000006755000001105 66C0110550C00000000
            VSEP                                                 QL.VSEP       0048
444444444444EECD444444444444444444444444444444444444444444DD4EECD44444
000000000000525700000000000000000000000000000000000000000083B525700000
                                      QL.VSEP                                  0090
4444444444444444444444444444444444444DD4EECD444444444444444444444444444444
000000000000000000000000000000000000083B525700000000000000000000000000000

Information displayed.
    5686-A06 (C) Copyright IBM Corp. 1998,2002  All Rights Reserved.
 ENTER = Process  PF2 = Main Menu    PF3 = Quit   PF4 = Next    PF5 = Prior
 PF7 = Up     PF8 = Down    PF9=Hex/Char   PF10=Txt/Head        PF12 = Monitor
```

Figure 57. API monitor - hexadecimal format

The layout of the displayed message is as follows:

```
01  MQ-MONITOR-RECORD.
   05  MQ-MON-TRAN-ID          PIC  X(4)  VALUE SPACES.
   05  MQ-MON-TERM-ID          PIC  X(4)  VALUE SPACES.
   05  MQ-MON-CICS-TASKN       PIC S9(8)  COMP VALUE ZERO.
   05  FILLER                  PIC  X(12) VALUE SPACES.
```

```
05  MQ-MON-SYSTEM-NUM       PIC  99    VALUE 01.
05  FILLER                  PIC  X(14) VALUE SPACES.
05  MQ-MON-FUNCTION         PIC  X(4)  VALUE SPACES.
   88 MQ-MON-CONNECT                VALUE 'CONN', 'CONI'
                                       'MCCO'.
   88 MQ-MON-CONNECT-VIA-APPL       VALUE 'CONN', 'CONI'.
   88 MQ-MON-CONNECT-VIA-INTERFACE VALUE 'CONI'.
   88 MQ-MON-MCP-CONNECT            VALUE 'MCCO'.
   88 MQ-MON-OPEN                   VALUE 'OPEN'.
   88 MQ-MON-PUT                    VALUE 'PUT '.
   88 MQ-MON-INQ                    VALUE 'INQ '.
   88 MQ-MON-GET                    VALUE 'GET '.
   88 MQ-MON-CLOSE                  VALUE 'CLOS'.
   88 MQ-MON-DISCONNECT             VALUE 'DISC'.

05  FILLER                  PIC  X(4)   VALUE SPACE.
05  MQ-MON-START-ABSTIME    PIC S9(15) COMP-3 VALUE ZERO.
05  MQ-MON-END-ABSTIME      PIC S9(15) COMP-3 VALUE ZERO.
05  MQ-MON-RESULTS.
    10  MQ-MON-CCODE        PIC S9(8)  COMP VALUE ZERO.
    10  MQ-MON-RCODE        PIC S9(8)  COMP VALUE ZERO.

05  FILLER                  PIC  X(12) VALUE SPACES.
05  MQ-MON-FUNCTION-DEP-INFO  VALUE SPACES.
    10  MQ-MON-QM-NAME      PIC  X(48).
    10  MQ-MON-Q-NAME       PIC  X(48).
    10  MQ-MON-RESOLVED-Q   PIC  X(48).
    10  FILLER              PIC  X(200).
```

It is possible to follow the flow of MQSeries API calls using a specific application. The application is identified by its CICS transaction code, terminal identifier, and CICS task number.

The specific MQSeries API calls are identified, together with the queue manager name, queue name, and condition and return codes.

Ensure that you toggle the MQSeries API monitor **off** after use.

Other areas of investigation

Perhaps the preliminary checks have enabled you to find the cause of the problem. If so, you should now be able to resolve it, possibly with the help of other books in the MQSeries library (see “Bibliography” on page 521) and in the libraries of other licensed programs.

If you have not yet found the cause, you must start to look at the problem in greater detail.

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

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

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

Have you obtained incorrect output?

In this book, "incorrect output" refers to your application:

- Not receiving a message that it was expecting
- Receiving a message containing unexpected or corrupted information
- Receiving a message that it was not expecting, for example, one that was destined for a different application

In all cases, check that any queue or queue manager aliases that your applications are using are correctly specified and accommodate any changes that have been made to your network.

If an MQSeries error message is generated, all of which are prefixed with the letters "MQI", you should look in the system log. See "System log" on page 159 for further information.

Does the problem occur at specific times of the day?

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

Is the problem intermittent?

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

Have you applied any service updates?

If a service update has been applied to MQSeries, check that the update action completed successfully and that no error message was produced.

- Did the update have any special instructions?
- Was any test run to verify that the update had been applied correctly and completely?
- Does the problem still exist if MQSeries is restored to the previous service level?
- If the installation was successful, check with the IBM Support Center for any patch error.
- If a patch has been applied to any other program, consider the effect it might have on the way MQSeries interfaces with it.

Does the problem affect only remote queues?

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

- Check that required channels have been started and are triggerable.
- Check that the programs that should be putting messages to the remote queues have not reported problems.
- If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on.

- Check the system log and VSE console for messages indicating channel errors or problems.

See the *MQSeries Intercommunication* book for information about how to define channels.

Is your application or MQSeries for VSE/ESA running slowly?

MQSeries for VSE/ESA runs as a subsystem, with a CICS partition, on the VSE operating system. The VSE operating system itself may be a second-level client on a VM machine. This complexity means that a performance problem can exist in any of these components.

MQSeries for VSE/ESA is sensitive to the CICS environment and availability of CICS resources. CICS performance problems are a specialized area requiring detailed analysis. Investigate these problems with the assistance of your CICS systems programmer.

MQSeries for VSE/ESA utilizes VSAM files under VSE. After prolonged use these files tend to fragment into several VSAM extents. This can be viewed with the VSE ICCF File and Catalog Management facility. Any files that show multiple extents should be reallocated with IDCAMS as soon as it is convenient to do so.

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

This could also be caused by a performance problem. Perhaps it is because your system is operating near the limits of its capacity.

Operating system performance problems, for both VSE/ESA and VM, are a specialized area requiring detailed analysis. Investigate these problems with the assistance of your VSE/ESA or VM systems programmer.

A performance problem may be caused by a limitation of your hardware.

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

The following symptoms might indicate that MQSeries is running slowly:

- Your system is slow to respond to MQSeries commands.
- Repeated displays of the queue depth indicate that the queue is being processed slowly for an application with which you would expect a large amount of queue activity.

If the performance of your system is still degraded after reviewing the above possible causes, the problem may lie with MQSeries for VSE/ESA itself. If you suspect this, you need to contact your IBM Support Center for assistance.

Application design considerations

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

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

Effect of message length

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

Searching for a particular message

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

Queues that contain messages of different lengths

If the messages on a queue are of different lengths, to determine the size of a message, your application could use the MQGET call with the *BufferLength* field set to zero so that, even though the call fails, it returns the size of the message data. The application could then repeat the call, specifying the identifier of the message it measured in its first call and a buffer of the correct size. However, if there are other applications serving the same queue, you might find that the performance of your application is reduced because its second MQGET call spends time searching for a message that another application has retrieved in the time between your two calls.

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

Use of the MQPUT1 call

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

Incorrect output

The term “incorrect output” can be interpreted in many different ways. For the purpose of problem determination within this book, the meaning is explained in “Have you obtained incorrect output?” on page 154.

Two types of incorrect output are discussed in this section:

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

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

Messages that do not appear on the queue

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

- Has the message been put on the queue successfully?
  - Has the queue been defined correctly. For example, are the queue and maximum message length sufficiently large?
  - Is the queue enabled for putting?
  - Is the queue already full? This could mean that an application was unable to put the required message on the queue.
- Are you able to get any messages from the queue?
  - Do you need to take a syncpoint?

    If messages are being put or retrieved within syncpoint, they are not available to other tasks until the unit of recovery has been committed.
  - Is your wait interval long enough?

    You can set the wait interval as an option for the MQGET call. You should ensure that you are waiting long enough for a response.
  - Are you waiting for a specific message that is identified by a message or correlation identifier (*MsgId* or *CorrelId*)?

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

    Also, check whether you can get other messages from the queue.
  - Can other applications get messages from the queue?
  - Has another application got exclusive access to the queue?

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

- Did the application get started?

  If it should have been triggered, check that the correct trigger options were specified.
- Did the application stop?
- Did the application complete correctly?

  Look for evidence of an abnormal end on the system log and VSE/ESA console.
- Did the application commit its changes, or were they backed out?

If multiple transactions are serving the queue, they can conflict with one another. For example, suppose one transaction issues an MQGET call with a buffer length of zero to find out the length of the message, and then issues a specific MQGET call specifying the *MsgId* of that message. However, in the meantime, another transaction issues a successful MQGET call for that message, so the first application receives a reason code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multi-server environment must be designed to cope with this situation.

Consider that the message could have been received, but that your application failed to process it in some way. For example, did an error in the expected format

of the message cause your program to reject it? If this is the case, refer to "Messages that contain unexpected or corrupted information."

Messages that contain unexpected or corrupted information

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

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

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

  For example, the format of the message data may have been changed, in which case both applications must be recompiled to pick up the changes. If one application has not been recompiled, the data will appear corrupt to the other.
- Is an application sending messages to the wrong queue?

  Check that the messages your application is receiving are not really intended for an application servicing a different queue.

  If your application has used an alias queue, check that the alias points to the correct queue.
- Has the trigger information been specified correctly for this queue?

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

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

Problems with incorrect output when using distributed queues

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

- Has MQSeries been correctly installed on both the sending and receiving systems, and correctly configured for distributed queuing?
- Are the links available between the two systems?

  Check that both systems are available, and connected to MQSeries. Check that the connection between the two systems, and the channels between the two queue managers, are active.
- Is triggering set on in the sending system?
- Is the message you are waiting for a reply message from a remote system?

  Check that triggering is activated in the remote system.
- Is the queue already full?

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

  The dead-letter queue header contains a reason or feedback code explaining why the message could not be put onto the target queue. See the *MQSeries Application Programming Reference* manual for information about the dead-letter queue header structure.
- Is there a mismatch between the sending and receiving queue managers?

  For example, the message length could be longer than the receiving queue manager can handle.
- Are the channel definitions of the sending and receiving channels compatible?

For example, a mismatch in sequence number wrap stops the distributed queuing component. See the *MQSeries Intercommunication* book for more information about distributed queuing.

System log

MQSeries uses the SYSTEM.LOG queue defined in the global system definition as its primary message log and additional informational messages are output to the VSE/ESA console. Typically, these detail starting, stopping, and initializing MQSeries for VSE/ESA

If the SYSTEM.LOG queue is unavailable, the messages are directed to the CICS CSMT log. These messages should always be reviewed carefully for any error messages. The type of messages included in the SYSTEM.LOG queue can now be controlled by using the 'Log and Trace Settings'. Refer to "Queue Manager Log and Trace Settings" on page 72 for details.

Dead-letter queues

Messages that cannot be delivered for some reason are placed on the dead-letter queue. You can check whether the queue contains any messages by using the MQMT transaction. If the queue contains messages, you can use the browse facility to browse messages on the queue using the MQGET call.

You must decide how to dispose of any messages found on the dead-letter queue, depending on the reasons for the messages being put on the queue.

Problems may occur if you do not have a dead-letter queue on each queue manager you are using.

Using MQSeries trace

MQSeries for VSE/ESA relies on the CICS auxiliary trace for problem determination. To reduce overhead in a production environment, the trace points are not issued unless specified using the 'Log and Trace Settings' screen. Tracing should only be used when requested by IBM service personnel. Refer to "Queue Manager Log and Trace Settings" on page 72 for details.

Problem determination with clients

An MQI client application receives MQRC_* reason codes in the same way as non-client MQI applications. However, there are now additional reason codes for error conditions associated with clients. For example:

- Remote machine not responding
- Communications line error
- Invalid machine address

The most common time for errors to occur is when an application issues an MQCONN and receives the response MQRC_Q_MQR_NOT_AVAILABLE. An error message, written to the client log file, explains the cause of the error. Messages may also be logged at the server depending on the nature of the failure.

Terminating clients

Even though a client has terminated it is still possible for the process at the server to be holding its queues open. Normally, this will only be for a short time until the communications layer detects that the partner has gone.

Error messages with clients

When an error occurs with a client system, error messages are put into the error files associated with the server, if possible. If an error cannot be placed there, the client code attempts to place the error message in an error log in the root directory of the client machine.

OS/2 and UNIX systems clients

Error messages for OS/2® and UNIX systems clients are placed in the error logs on their respective MQSeries server systems. Typically, these files appear in the QMGRS\@SYSTEM\ERRORS directory.

DOS and Windows clients

The location of the log file AMQERR01.LOG is set by the MQDATA environment variable. The default location, if not overridden by MQDATA, is:

```
C:\
```

Working in the DOS environment involves the environment variable MQDATA.

This is the default library used by the client code to store trace and error information; it also holds the directory name in which the qm.ini file is stored. (needed for NetBIOS setup). If not specified, it defaults to the C drive.

The names of the default files held in this library are:

AMQERR01.LOG
For error messages.

AMQERR01.FDC
For First Failure Data Capture messages.

Problems with SSL enabled channels

SSL enabled channels can fail for a number of reasons. Sometimes a channel will fail because it is meant to fail. That is, the remote system provided an X.509 certificate that did not match expected identifiers. Specifically, the partner certificate failed to match the SSL Peer Attributes parameter of the local channel.

In this case, a message is logged to the SYSTEM.LOG and the channel is terminated. This is the correct behavior in such a situation, as the channel should not continue when the remote system does not identify itself as expected.

SSL enabled channels can also fail due to problems with the MQSeries, VSE/ESA or the remote system environment. Specifically, an SSL enabled channel can fail due to:

- SSL availability
- Cipher specification support
- Client authentication failure
- General channel failure

SSL availability

It is essential for SSL enabled channels that the SSL feature is installed and available on both the local and remote systems participating in an SSL secured conversation.

MQSeries for VSE/ESA initializes the SSL environment at system startup, if the an SSL key-ring sublibrary name is specified in the queue manager's communication settings. At startup, MQSeries will create an initialization instance for use by channels during normal system operation. The initialization instance is a data structure provided by the SSL service, and is associated with the SSL key-ring sublibrary.

The SSL initialization instance is anchored to an MQSeries control block and is used each time an SSL enabled channel is activated. If the instance is invalid, all SSL enabled channels will fail during initialization, with the following error message:

```
006100E TCP/IP SSL INITIALIZATION FAILED
```

This particular error can also occur if the SSL key-ring member name does not identify valid private key and certificate files in the SSL key-ring sublibrary.

In addition, and in regard to the remote system, this error can occur if the remote system does not have the SSL feature installed and available, or the cipher specification required by the SSL client is not supported by the SSL server (that is, the receiver MCA).

If this message is encountered, the following should be checked:

- The SSL feature is installed and available.
- The SSL key-ring sublibrary name specified in the queue manager 'Communication Settings' identifies the correct sublibrary name.
- The SSL key-ring member name, also specified in the queue manager 'Communication Settings' identifies the member name of valid private key and certificate files.
- The remote system has SSL installed and available.
- The remote system supports the cipher specification identified by Sender channel.
- The remote system is configured to provide an X.509 certificate during SSL initial negotiation.

Following these checks, if the channel continues to fail, problem determination relating to channels in general should be pursued.

Cipher specification support

AN SSL enabled channel can fail if the SSL Cipher Specification identified by the Sender channel is not supported by the remote system, or if the SSL negotiations complete for a specification that is not identified by the Receiver channel.

Both of these cases produce the following error message:

```
006101E TCP/IP SSL CIPHER SPECIFICATION ERROR
```

For Sender channels, it is important that the cipher specification identified in the channel definition is supported by the remote system. The TCP/IP Administrator of the remote system should be able to provide a list of supported ciphers. For MQSeries for VSE/ESA, the cipher is identified by a two-character code, (for

example, 08, 09, 0A, 62). SSL for VSE documentation should be reviewed to determine what ciphers these codes identify and whether or not the remote system can support them.

For Receiver channels, it is essential that the cipher identified in the channel definition matches the cipher stipulated by the matching Sender channel definition. For example, if the Sender channel specifies a cipher code of '0A', then the Receiver channel must also specify '0A'. For Receiver channels, MQSeries completes the SSL negotiation and then checks that the agreed cipher matches the channel definition. If not, the channel is terminated.

Client authentication failure

The SSL client authentication parameter, specified on a channel definition, instructs MQSeries to check that the remote partner exchanged an X.509 certificate during SSL negotiation. SSL negotiation occurs when the channel is initialized.

If a channel is configured to require a certificate from the remote system and one is not received, then a client authentication error will occur. Since the server, or MQSeries receiver MCA, always provides a certificate during SSL negotiation, this error pertains to Sender channels only.

A client authentication failure produces the following error message:

```
006103E TCP/IP SSL CLIENT AUTHENTICATION ERROR
```

However, a client authentication error cannot occur with the current SSL for VSE service, because MQSeries for VSE/ESA always requires a certificate from the client during SSL negotiation. If a certificate is not exchanged, MQSeries will terminate the channel with an SSL initialization error, not a client authentication error.

General channel failure

It is possible for an SSL enabled channel to successfully establish an active SSL connection with a remote system and subsequently fail. In such a case, the channel has probably failed for any of the reasons applicable to channels in general.

When a TCP/IP connection between two systems (or between the queue manager and a remote client program) is established, MQSeries will attempt to secure the connection using SSL services if the Sender (or client) channel is SSL enabled. This process may be successful, but subsequent channel negotiations over the secure connection fail. This is not an SSL channel failure.

If a channel fails due to reasons applicable to channels in general, problem determination relevant to general channel failure should be pursued.

Chapter 7. Message data conversion

Application data is converted within an application program when the MQGMO-CONVERT option is specified in the Options field of the MQGMO structure passed to an MQGET call, and all of the following are true:

- The code page or encoding fields set in the MQMD structure associated with the message on the queue differ from the code page or encoding fields set in the MQMD structure specified on the MQGET call.
- The format field in the MQMD structure associated with the message is not MQFMT-NONE.
- The buffer length specified on the MQGET call is not zero.
- The message data length is not zero.
- The queue manager supports conversion between the code page and encoding fields specified in the MQMD structures associated with the message and the MQGET call. LE/VSE is used for application data conversion and must have the appropriate code converters available. See "Using LE/VSE for conversion" on page 164 for more information.

The queue manager supports conversion of a number of data formats. If the format field of the MQMD structure associated with the message is one of the built-in formats, the queue manager can convert the message. If the format is not one of the built-in formats, you must write a data conversion exit program to convert the message.

When you move messages between systems, sometimes it is necessary to convert the application data into the character set and encoding required by the receiving system. Conversion can be done either from within application programs on the receiving system, or by the Message Channel Agents (MCAs) on the sending system. If data conversion is supported on the receiving system, we recommend that you use application programs to convert the application data, rather than depending on the data already being converted at the sending system.

If the sending MCA is to convert the data, the 'Convert Msgs' field must be set to Y on the definition of each sender channel for which conversion is required. If conversion fails on the sender channel, the message remains on the transmission queue and the channel is shut down, with a GET error shown on the system log.

The following built-in formats are converted by MQSeries for VSE/ESA:

MQFMT_STRING
MQFMT_DEAD_LETTER_HEADER
MQFMT_TRIGGER
MQFMT_ADMIN
MQFMT_EVENT
MQFMT_PCF
MQFMT_REF_MSG_HEADER
MQFMT_IMS
MQFMT_IMS_VAR_STRING
MQFMT_COMMAND_1
MQFMT_COMMAND_2
MQFMT_SAP
MQFMT_RF_HEADER
MQFMT_RF_HEADER_2

© Copyright IBM Corp. 1995, 2004

MQFMT_CICS
MQFMT_DIST_HEADER

See Appendix D of the *MQSeries Application Programming Reference* manual for more details regarding data conversion.

Data conversion exit programs

For application defined formats that do not conform to the built-in formats, conversion can be performed by an exit program.

The name of the exit program must be the same as your data format name. For example, if a message has the format (FMT_TEST), the exit program must be called FMT_TEST. MQSeries checks that the format is not one of the built-in formats, then, if it is not, calls the exit program.

To build a user exit program for your own format:

1. Start with the supplied source skeleton DCHFMT4.
2. Follow the instructions in the prolog of DCHFMT4, ensuring that the correct macros are called to convert your structure.
3. Rename the program to your data format name.
4. CICS translate, compile, and link-edit your program.
5. Place your exit program in a CICS application program library. Define the program to CICS in the usual way.

See Chapter 11 of the *MQSeries Application Programming Guide* for more details regarding data conversion exit programs.

Using LE/VSE for conversion

For application message code page conversion, MQSeries for VSE/ESA uses the Language Environment® (LE) code set conversion facilities in a similar manner to the MQSeries server. LE/VSE provides a number of supplied code converters, and facilities to build code converters that are not provided.

If you need code conversion for pages that are not provided by LE/VSE, you can edit the appropriate source code modules and build the converters. You also need to inform MQSeries for VSE/ESA of the type number and the encoding of the user-defined code page.

Selecting 4 on the configuration menu allows you to add user-defined code pages, and their type and encoding.

As well as the MQServer SBCS conversion, support is provided for:

- DBCS code pages
- Mixed code pages
- EUC code pages
- ISO code page
- Unicode (UCS-2 and UTF-8) code pages

See “Client code-page conversion tables” on page 465 for more details on LE/VSE code pages. The same LE code set conversion facilities are used for application data conversion.

We also strongly recommended that you read the section on code page conversion in the *C Run-time Programming Guide*.

Note: The following LE code pages have the equivalent numerics on MQSeries for VSE/ESA.

ISO8859-1 - 819

IOS8859-7 - 813

ISO8859-9 - 920

IBM-eucJP - 33722

Building a conversion exit program

MQSeries for VSE/ESA is shipped with a sample exit program (DCHFMT4). This sample provides conversion for standard data types recognized by supplied conversion macros. The following data structure encapsulates these supported data types:

```
struct testall
{
  MQBYTE  byte2[2];
  short   short2;
  MQLONG  long4;
  MQCHAR  char5[5];
  MQCHAR  charV;
};
```

This contains all the types of data (char, variable char, byte, long and short) that can be converted.

The following code in DCHFMT4 converts the structure shown earlier.

```
ConvertByte(2);     /* 3 byte binary data              */

AlignShort();
ConvertShort(1);    /* 2 byte integer                  */

AlignLong();
ConvertLong(1);     /* 4 byte integer                  */

ConvertChar(5);     /* 5 byte character data           */

ConvertVarChar();   /* Variable length character data  */
```

Note the use of AlignShort/Long before ConvertShort/Long. In VSE, there is no CRTMQCVX utility to create this code fragment for you.

These functions are defined using the following include files (see the DCHFMT4 source for a full listing):

```
#include <cmqc.h>            /* For MQI datatypes
#include <cmqxc.h>           /* For MQI exit-related definitions
#include <mqidatcv.h>        /* For sample macro definitions
```

The following linkage parameters are required to build a conversion exit program:

```
PHASE DCHFMT4,*
INCLUDE DCHFMT4
INCLUDE MQPDATCU
INCLUDE MQPDATCV
INCLUDE DFHELII
```

Because the exit program runs under CICS, a PPT entry is required in CICS for the program. For example:

```
DEFINE  PROGRAM(DCHFMT4) GROUP(MQM) LANGUAGE(C)
```

Note: The user exit program must be contained in the same CICS region as MQSeries — it cannot be placed in a separate region for dynamic routing.

Chapter 8. Programmable system management

This chapter describes MQSeries PCFs (Programmable Command Formats) and their relationship to other parts of the MQSeries products.

The PCFs described in this chapter are supported by:

- MQSeries for AIX®
- MQSeries for AS/400
- MQSeries for AT&T GIS UNIX
- MQSeries for Digital OpenVMS
- MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX)
- MQSeries for HP-UX
- MQSeries for OS/2 Warp
- MQSeries for SINIX and DC/OSx
- MQSeries for Sun Solaris
- MQSeries for Tandem NonStop Kernel
- MQSeries for Windows NT®
- MQSeries for Windows® Version 2 Release 1

Introduction to Programmable Command Formats (PCFs)

The problem PCF commands solve

The administration of distributed networks can become very complex. The problems of administration will continue to grow as networks increase in size and complexity.

Examples of administration specific to messaging and queuing include:

- Resource management.

  For example, queue creation and deletion.
- Performance monitoring.

  For example, maximum queue depth or message rate.
- Control.

  For example, tuning queue parameters such as maximum queue depth, maximum message length, and enabling and disabling queues.
- Message routing.

  Definition of alternative routes through a network.

MQSeries PCF commands can be used to simplify queue manager administration and other network administration. PCF commands allow you to use a single application to perform network administration from a single queue manager within the network.

What PCFs are

PCFs define command and reply messages that can be exchanged between a program and any queue manager (that supports PCFs) in a network. You can use PCF commands in a systems management application program for administration

© Copyright IBM Corp. 1995, 2004

of MQSeries objects: queue managers, process definitions, queues, and channels. The application can operate from a single point in the network to communicate command and reply information with any queue manager, local or remote, via the local queue manager.

Each queue manager has an administration queue with a standard queue name and your application can send PCF command messages to that queue. Each queue manager also has a command server to service the command messages from the administration queue. PCF command messages can therefore be processed by any queue manager in the network and the reply data can be returned to your application, using your specified reply queue. PCF commands and reply messages are sent and received using the normal Message Queue interface (MQI).

Preparing MQSeries for PCF

MQSeries for VSE/ESA requires several resources to be active and available before it can support PCFs. Specifically, MQSeries requires:

- System command queue is defined and available.
- PCF command server is active in CICS.

System command queue

The system command queue (usually SYSTEM.ADMIN.COMMAND.QUEUE) is specified in the queue manager's global system definition, as a communication parameter.

The global system definition can be accessed using the MQMT transaction, option 1.1 (the system command queue is one of the queue manager's "Communication Settings" and is accessible using PF9):

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
 14:47:42                  Global System Definition                   CIC1
 MQMMSYS                    Communications Settings                   A000

     TCP/IP settings                          Batch Interface settings
     TCP/IP listener port : 01414             Batch Int. identifier: MQBISERV
     Licensed clients . . : 00000             Batch Int. auto-start: Y
     Adopt MCA  . . . . . : Y
     Adopt MCA Check  . . : Y

     SSL parameters
     Key-ring sublibrary  : PRD2.SSLKEYS
     Key-ring member  . . : MQVSEKEY

     PCF parameters
     System command queue : SYSTEM.ADMIN.COMMAND.QUEUE
     System reply queue . : SYSTEM.ADMIN.REPLY.QUEUE
     Cmd Server auto-start: Y
     Cmd Server convert . : N
     Cmd Server DLQ store : Y

 Requested record displayed.
 PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 58. PCF parameters

PCF messages, to be processed by the VSE queue manager, should be placed on the queue identified by the system command queue parameter (the system reply queue parameter is used by the MQSC batch utility, and is not relevant in the current context).

PCF messages placed on the system command queue are processed by the PCF command server, transaction MQCS.

PCF command server

The PCF command server must be active in CICS before PCF messages can be processed by MQSeries for VSE/ESA. The PCF command server (transaction MQCS) can be activated in two ways:

- Manually in native CICS.
- Automatically by the queue manager.

Starting the command server manually

The PCF command server can be manually started in native CICS by running the MQCS transaction. If successful, starting the command server this way will display the following message:

```
PCF Command Server started.
```

The following message will also be written to the MQSeries system log (assuming the queue manager is configured to log informational messages, see "Queue Manager Log and Trace Settings" on page 72.

```
MQI007000I PCF command server started
```

If the command server fails to start, review the system log for associated error messages.

Stopping the command server manually

The PCF command server can be manually stopped in native CICS by running the MQCS transaction with the terminate ("X") parameter. For example:

```
MQCS X
```

The terminate parameter places a special stop request message on the system command queue. The command server, when it retrieves the stop request from the command queue, finishes processing and terminates. If successful, running MQCS with the terminate parameter will display the following message:

```
PCF Command Server termination requested.
```

If the stop request is successful, the command server stops and the following message is written to the system log:

```
MQI007017I PCF command server stopped
```

If the stop request fails, or the command server fails to stop, review the system log for associated error messages.

Starting the command server automatically

The PCF command server can be started automatically in CICS by configuring the queue manager to start the server when the queue manager is initialized. This is achieved by setting the command server auto-start option on.

The command server auto-start parameter is configurable as part of the queue manager's global system definition (MQMT option 1.1) and the communication settings (PF9). See Figure 58 on page 168.

The command server auto-start parameter can be set to (Y)es or (N)o. To activate the command server automatically when the queue manager is initialized, set the auto-start parameter to (Y)es.

The command server runs as transaction MQCS. The auto-activation of the command server can be verified by inquiring on the active tasks in CICS. If the command server is active, the MQCS transaction will be listed as an active task.

If the command server fails to start automatically, check the system log for associated error messages.

Stopping the command server automatically

The command server can be stopped automatically by stopping the queue manager (by running the MQST transaction).

The command server can be stopped automatically this way regardless of how the server was started. For example, if the command server was started manually in native CICS, it can still be stopped automatically by stopping the queue manager.

Using PCFs

This section describes how to use the PCFs in a systems management application program for MQSeries remote administration. The topic includes:

- PCF command messages
- Responses
- Authority checking for PCF commands

PCF command messages

Each command and its parameters are sent as a separate command message containing a PCF header followed by a number of parameter structures (see "MQCFH - PCF header" on page 240). The PCF header identifies the command and the number of parameter structures that follow in the same message. Each parameter structure provides a parameter to the command.

Replies to the commands, generated by the command server, have a similar structure. There is a PCF header, followed by a number of parameter structures. Replies can consist of more than one message but commands always consist of one message only.

The queue to which the PCF commands are sent is often called the SYSTEM.ADMIN.COMMAND.QUEUE. The command server servicing this queue sends the replies to the queue defined by the ReplyToQ and ReplyToQMgr fields in the message descriptor of the command message.

How to issue PCF command messages

Use the normal Message Queue Interface (MQI) calls, MQPUT, MQGET and so on, to put and retrieve PCF command and response messages to and from their respective queues.

PCF commands should be placed on the system command queue (specified in the queue manager's global system definition). PCF response messages are place on the ReplyToQ specified in the message descriptor of the originating PCF command.

Message descriptor for a PCF command

The MQSeries message descriptor is fully documented in the MQSeries Application Programming Reference manual.

A PCF command message contains these fields in the message descriptor:

Report
Any valid value, as required.

MsgType
This must be MQMT_REQUEST to indicate a message requiring a response.

Expiry Any valid value, as required.

Feedback
Set to MQFB_NONE

Encoding
If you are sending to AS/400, OS/2, Windows NT, or UNIX systems, set this field to the encoding used for the message data; conversion will be performed if necessary.

CodedCharSetId
If you are sending to AS/400, OS/2, Windows NT, or UNIX systems, set this field to the coded character-set identifier used for the message data; conversion will be performed if necessary.

Format
Set to MQFMT_ADMIN.

Priority
Any valid value, as required.

Persistence
Any valid value, as required.

MsgId The sending application may specify any value, or MQMI_NONE can be specified to request the queue manager to generate a unique message identifier.

CorrelId
The sending application may specify any value, or MQCI_NONE can be specified to indicate no correlation identifier.

ReplyToQ
The name of the queue to receive the response.

ReplyToQMgr
The name of the queue manager for the response (or blank).

Message context fields
These can be set to any valid values, as required. Normally the Put message option MQPMO_DEFAULT_CONTEXT is used to set the message context fields to the default values.

If you are using a version-2 MQMD structure, you must set these additional fields:

GroupId
Set to MQGI_NONE

MsgSeqNumber
Set to 1

Offset Set to 0

MsgFlags
Set to MQMF_NONE

OriginalLength
Set to MQOL_UNDEFINED

Sending user data

MQSeries for VSE/ESA does not support user-defined PCF formats. Unlike some MQ platforms that support PCF messages with an MQMD format of MQFMT_PCF, the MQSeries for VSE/ESA command server will reject any PCF message that does not have an MQMD Format of MQFMT_ADMIN.

Responses

In response to each command, the command server generates one or more response messages. A response message has a similar format to a command message; the PCF header has the same command identifier value as the command to which it is a response (see "MQCFH - PCF header" on page 240 for details). The message identifier and correlation identifier are set according to the report options of the request.

If a single command specifies a generic object name, a separate response is returned in its own message for each matching object. For the purpose of response generation, a single command with a generic name is treated as multiple individual commands (except for the control field MQCFC_LAST or MQCFC_NOT_LAST). Otherwise, one command message generates one response message.

There are three types of response, described below:

- OK response
- Error response
- Data response

OK response

This consists of a message starting with a command format header, with a CompCode field of MQCC_OK or MQCC_WARNING.

For MQCC_OK, the Reason is MQRC_NONE.

For MQCC_WARNING, the Reason identifies the nature of the warning. In this case the command format header may be followed by one or more warning parameter structures appropriate to this reason code.

In either case, for an inquire command further parameter structures may follow.

Error response

If the command has an error, one or more error response messages are sent (more than one may be sent even for a command which would normally only have a single response message). These error response messages have MQCFC_LAST or MQCFC_NOT_LAST set as appropriate.

Each such message starts with a response format header, with a CompCode value of MQCC_FAILED and a Reason field which identifies the particular error. In general each message describes a different error. In addition, each message has

either zero or one (never more than one) error parameter structures following the header. This parameter structure, if there is one, is an MQCFIN structure, with a Parameter field containing one of:

MQIACF_PARAMETER_ID
The Value field in the structure is the parameter identifier of the parameter that was in error (for example, MQCA_Q_NAME).

MQIACF_ERROR_ID
This is used with a Reason value (in the command format header) of MQRC_UNEXPECTED_ERROR. The Value field in the MQCFIN structure is the unexpected reason code received by the command server.

MQIACF_SELECTOR
This occurs if a list structure (MQCFIL) sent with the command contains an invalid or duplicate selector. The Reason field in the command format header identifies the error, and the Value field in the MQCFIN structure is the parameter value in the MQCFIL structure of the command that was in error.

MQIA_CODED_CHAR_SET_ID
This occurs when the coded character-set identifier in the message descriptor of the incoming PCF command message does not match that of the target queue manager. The Value field in the structure is the coded character-set identifier of the queue manager.

The last (or only) error response message is a summary response, with a CompCode field of MQCC_FAILED, and a Reason field of MQRCCF_COMMAND_FAILED. This message has no parameter structure following the header.

Data response

This consists of an OK response (as described above) to an inquire command. The OK response is followed by additional structures containing the requested data.

Applications should not depend upon these additional parameter structures being returned in any particular order.

For specific information about data response messages, refer to "Data responses to commands" on page 226.

Message descriptor for a response

A response message (obtained using the Get-message option MQGMO_CONVERT) has the following fields in the message descriptor, defined by the putter of the message. The actual values in the fields are generated by the queue manager:

MsgType
This is MQMT_REPLY.

MsgId This is generated by the queue manager.

CorrelId
This is generated according to the report options of the command message.

Format
This is MQFMT_ADMIN.

Encoding
Set to MQENC_NATIVE.

CodedCharSetId
Set to MQCCSI_Q_MGR.

Persistence
The same as in the command message.

Priority
The same as in the command message.

The response is generated with MQPMO_PASS_IDENTITY_CONTEXT.

Authority checking for PCF commands

When a PCF command is processed, the UserIdentifier from the message descriptor in the command message is used for the required MQSeries object authority checks. The checks are performed on the system on which the command is being processed, therefore this user ID must exist on the target system and have the required authorities to process the command.

If the message has come from a remote system, one way of achieving this is to have a matching user ID on both the local and remote systems.

Authority checking is implemented differently on each platform. For more information about PCF command, and command resource authority checking, refer to Chapter 11, “Security,” on page 285.

Definitions of the PCFs

This section contains reference material for the PCFs of commands and responses sent between an MQSeries systems management application program and an MQSeries queue manager.

MQSeries for VSE/ESA supports these PCF commands:
- Change Channel
- Change Queue
- Change Queue Manager
- Copy Channel
- Copy Queue
- Create Channel
- Create Queue
- Delete Channel
- Delete Queue
- Escape
- Inquire Channel
- Inquire Channel Names
- Inquire Queue
- Inquire Queue Manager
- Inquire Queue Names
- Ping Queue Manager
- Reset Channel
- Start Channel
- Start Channel Listener
- Stop Channel

Programmable constants and data structures for these commands and their parameters are available in copybooks and header files provided with MQSeries for VSE/ESA as follows:

- CMQC.H
- CMQXC.H
- CMQCFC.H
- CMQV.C
- CMQXV.C
- CMQCFV.C

Note that the command descriptions in the following sections pertain to MQSeries for VSE/ESA. When considering PCF messages intended for other platforms, you should also refer to MQSeries Programmable System Management (SC33-1482-08).

Error codes applicable to all commands

In addition to those listed under each command format, any command may return the following in the response format header (descriptions of the MQRC_* error codes are given in the MQSeries Application Programming Reference manual):

Reason (MQLONG)

The value may be:

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Not authorized for access.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Message length greater than maximum for queue.

MQRC_NONE
(0, X'000') No reason to report.

MQRCCF_COMMAND_FAILED
Command failed.

MQRCCF_CFH_COMMAND_ERROR
Command identifier not valid.

MQRCCF_CFH_CONTROL_ERROR
Control option not valid.

MQRCCF_CFH_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFH_MSG_SEQ_NUMBER_ERR
Message sequence number not valid.

MQRCCF_CFH_PARM_COUNT_ERROR
Parameter count not valid.

MQRCCF_CFH_TYPE_ERROR
Type not valid.

MQRCCF_CFH_VERSION_ERROR
Structure version number is not valid.

MQRCCF_ENCODING_ERROR
Encoding error.

MQRCCF_MD_FORMAT_ERROR
Format not valid.

MQRCCF_MSG_TRUNCATED
Message truncated.

MQRCCF_MSG_LENGTH_ERROR
Message length not valid.

MQRCCF_MSG_SEQ_NUMBER_ERROR
Message sequence number not valid.

Change channel

The Change Channel (MQCMD_CHANGE_CHANNEL) command changes the specified attributes in a channel definition.

This PCF is supported on all platforms.

For any optional parameters that are omitted, the value does not change.

Required parameters:

ChannelName, ChannelType

Optional parameters:

AllocRetryCount, AllocRetryFastTimer, AllocRetrySlowTimer, BatchSize, ChannelDesc, ConnectionName, DataConversion, DiscInterval, DiscRetryCount, MaxMsgLength, MsgExit, MsgUserData, PortNumber, ReceiveExit, ReceiveUserData, SecurityExit, SecurityUserData, SendExit, SendUserData, SeqNumberWrap, SSLCipherSpec, SSLClientAuth, SSLPeerName, TpName, TransportType, XmitQName

Required parameters

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Specifies the name of the channel definition to be changed.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ChannelType (MQCFIN)
Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

Specifies the type of channel being changed. The value may be:

MQCHT_SENDER
Sender.

MQCHT_RECEIVER
Receiver.

MQCHT_SVRCONN
Client.

Optional parameters

AllocRetryCount (MQCFIN)
APPC connection retry count (parameter identifier: MQIACH_ALLOC_RETRY).

Specifies the number of times an LU 6.2 Sender connection will attempt to allocate a session before failing.

Specify a value in the range 0 through 99 999 999.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

AllocRetryFastTimer (MQCFIN)
APPC connection retry fast timer (parameter identifier: MQIACH_ALLOC_FAST_TIMER).

The time interval, in seconds, that an allocation of conversation is retried for the first cycle of retries. A value of one to five seconds is sufficient for this parameter, with the longer time being used for a slow environment, for example, a dial-up SDLC.

Specify a value in the range 0 through 99 999 999.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

AllocRetrySlowTimer (MQCFIN)
APPC connection retry slow timer (parameter identifier: MQIACH_ALLOC_SLOW_TIMER).

The time interval, in seconds, that an allocation of conversation is retried for a second cycle of retries, should the first (fast) cycle of retries fail. A value between 3 and 10 seconds is sufficient for this field, with the longer time being used for a slow environment.

Specify a value in the range 0 through 99 999 999.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

BatchSize (MQCFIN)
Batch size (parameter identifier: MQIACH_BATCH_SIZE).

The maximum number of messages that should be sent down a channel before a checkpoint is taken.

The batch size which is actually used is the lowest of:

- The BatchSize of the sending channel
- The BatchSize of the receiving channel

Specify a value in the range 1 through 999 999.

This parameter is not valid for channels with a ChannelType of MQCHT_SVRCONN.

ChannelDesc (MQCFST)
Channel description (parameter identifier: MQCACH_DESC).

The maximum length of the string is MQ_CHANNEL_DESC_LENGTH.

Use characters from the character set, identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing, to ensure that the text is translated correctly.

ConnectionName (MQCFST)
Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH.

Specify the name of the machine as required for the stated TransportType:

- For MQXPT_LU62, specify the fully-qualified name of the partner LU.
- For MQXPT_TCP specify either the host name or the network address of the remote machine.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

DataConversion (MQCFIN)
Whether sender should convert application data (parameter identifier: MQIACH_DATA_CONVERSION).

This parameter is valid only for ChannelType values of MQCHT_SENDER.

The value may be:

MQCDC_NO_SENDER_CONVERSION
No conversion by sender.

MQCDC_SENDER_CONVERSION
Conversion by sender.

DiscInterval (MQCFIN)
Disconnection interval (parameter identifier: MQIACH_DISC_INTERVAL).

This defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before terminating the channel. A value of zero causes the message channel agent to wait indefinitely.

Specify a value in the range 0 through 999 999.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

DiscRetryCount (MQCFIN)
Disconnection retry count (parameter identifier: MQIACH_DISC_RETRY).

Specifies the maximum number of retries that the channel waits for messages to be put on a transmission queue before terminating the channel. The retries occur at a frequency specified by the DiscInterval parameter.

Specify a value in the range 0 through 99 999 999.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

MaxMsgLength (MQCFIN)
Maximum message length (parameter identifier: MQIACH_MAX_MSG_LENGTH).

Specifies the maximum message length that can be transmitted on the channel. This is compared with the value for the remote channel and the actual maximum is the lowest of the two values.

The value zero means the maximum message length for the queue manager.

The lower limit for this parameter is 0. The upper limit depends on the environment. For MQSeries for VSE/ESA the upper limit is 4 MB.

MsgExit (MQCFST)
Message exit name (parameter identifier: MQCACH_MSG_EXIT_NAME).

If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

For channels with a channel type (ChannelType) of MQCHT_SVRCONN, this parameter is not relevant, since message exits are not invoked for such channels.

The format of the string is a 1-8 character CICS program name. The exit name must identify a program defined to the CICS region.

The maximum length of the exit name is MQ_EXIT_NAME_LENGTH.

MsgUserData (MQCFST)
Message exit user data (parameter identifier:

MQCACH_MSG_EXIT_USER_DATA).

Specifies user data that is passed to the message exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

PortNumber (MQCFIN)
TCP/IP port number (parameter identifier: MQIACH_PORT_NUMBER).

For TCP/IP channels only, specifies the port number (for example, 1414) corresponding to an MQ Listener process running on the host specified by the ConnectionName parameter.

This parameter is valid only for ChannelType values of MQCHT_SENDER.

ReceiveExit (MQCFST)
Receive exit name (parameter identifier: MQCACH_RCV_EXIT_NAME).

If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The format of the string is a 1-8 character CICS program name. The exit name must identify a program defined to the CICS region.

The maximum length of the exit name is MQ_EXIT_NAME_LENGTH.

ReceiveUserData (MQCFST)
Receive exit user data (parameter identifier: MQCACH_RCV_EXIT_USER_DATA).

Specifies user data that is passed to the receive exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

SecurityExit (MQCFST)
Security exit name (parameter identifier: MQCACH_SEC_EXIT_NAME).

If a nonblank name is defined, the security exit is invoked at the following times:

- Immediately after establishing a channel.
- Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- Upon receipt of a response to a security message flow. Any security message flows received from the remote processor on the remote machine are passed to the exit.

The format of the string is a 1-8 character CICS program name. The exit name must identify a program defined to the CICS region.

The maximum length of the exit name is MQ_EXIT_NAME_LENGTH.

SecurityUserData (MQCFST)
Message exit user data (parameter identifier: MQCACH_SEC_EXIT_USER_DATA).

Specifies user data that is passed to the security exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

SendExit (MQCFST)
Send exit name (parameter identifier: MQCACH_SEND_EXIT_NAME).

If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The format of the string is a 1-8 character CICS program name. The exit name must identify a program defined to the CICS region.

The maximum length of the exit name is MQ_EXIT_NAME_LENGTH.

SendUserData (MQCFST)
Send exit user data (parameter identifier: MQCACH_SEND_EXIT_USER_DATA).

Specifies user data that is passed to the receive exit.

The maximum length of the string is MQ_EXIT_DATA_LENGTH.

SeqNumberWrap (MQCFIN)
Sequence wrap number (parameter identifier: MQIACH_SEQUENCE_NUMBER_WRAP).

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

Specify a value in the range 100 through 999 999.

This parameter is not valid for channels with a ChannelType of MQCHT_SVRCONN.

SSLCipherSpec (MQCFST)
SSL cipher specification (parameter identifier: MQCACH_SSL_CIPHER_SPEC).

Specifies the SSL cipher specification to use when establishing an SSL enabled channel. Unlike othe MQ platforms that expect a 32-character specification name, MQSeries for VSE/ESA expects a 2-character specification code.

For a list of valid cipher specification codes, refer to SSL product documentation.

The maximum length of the string is MQ_SSL_CIPHER_SPEC_LENGTH.

SSLClientAuth (MQCFIN)
SSL client authentication (parameter identifier: MQIACH_SSL_CLIENT_AUTH).

Specifies whether or not a PKI certificate is required when establishing an SSL enabled channel.

The value may be:

MQSCA_REQUIRED
PKI certificate is required.

MQSCA_OPTIONAL
PKI certificate is optional.

SSLPeerName (MQCFST)
SSL peer name (parameter identifier: MQCACH_SSL_PEER_NAME)

Identifies certain characteristics that are expected to match the contents of a PKI certificate received when establishing an SSL enabled channel.

The maximum length of the string is MQ_DISTINGUISHED_NAME_LENGTH.

TpName (MQCFST)
Transaction program name (parameter identifier: MQCACH_TP_NAME).

This is the LU 6.2 transaction program name.

The maximum length of the string is MQ_TP_NAME_LENGTH.

This parameter is valid only for channels with a TransportType of MQXPT_LU62. It is not valid for receiver channels.

TransportType (MQCFIN)
Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

No check is made that the correct transport type has been specified if the channel is initiated from the other end. The value may be:

MQXPT_LU62
LU 6.2.

MQXPT_TCP
TCP/IP.

XmitQName (MQCFST)
Transmission queue name (parameter identifier: MQCACH_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

A transmission queue name is required (either previously defined or specified here) if ChannelType is MQCHT_SENDER. It is not valid for other channel types.

Note that the optional parameters described in this section are also applicable to the following PCF commands:

- "Copy Channel" on page 197
- "Create Channel" on page 202

Error codes

In addition to the values for any command shown in "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_BATCH_SIZE_ERROR
Batch size not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFSL_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFSL_TOTAL_LENGTH_ERROR
Total string length error.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_NAME_ERROR
Channel name error.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_DISC_INT_ERROR
Disconnection interval not valid.

MQRCCF_DISC_INT_WRONG_TYPE
Disconnection interval not allowed for this channel type.

MQRCCF_MAX_MSG_LENGTH_ERROR
Maximum message length not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_SEQ_NUMBER_WRAP_ERROR
Sequence wrap number not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

MQRCCF_XMIT_PROTOCOL_TYPE_ERR
Transmission protocol type not valid.

MQRCCF_XMIT_Q_NAME_ERROR
Transmission queue name error.

MQRCCF_XMIT_Q_NAME_WRONG_TYPE
Transmission queue name not allowed for this channel type.

Change queue

The Change Queue (MQCMD_CHANGE_Q) command changes the specified attributes of an existing MQSeries queue.

This PCF is supported on all platforms.

For any optional parameters that are omitted, the value does not change.

Required parameters:

QName, QType

Optional parameters:

BaseQName, CICSFileName, InhibitGet, InhibitPut, MaxGlobalLocks, MaxLocalLocks, MaxMsgLength, MaxQDepth, MaxQTriggers, MaxQUsers, QDepthHighEvent, QDepthHighLimit, QDepthLowEvent, QDepthLowLimit, QDepthMaxEvent, QDesc, QServiceInterval, QServiceIntervalEvent, RemoteQMgrName, RemoteQName, Shareability, TriggerChannelName, TriggerControl, TriggerData, TriggerProgramName, TriggerRestart, TriggerTerminalId, TriggerTransactionId, TriggerType, Usage, XmitQName

Required parameters

QName (MQCFST)
Queue name (parameter identifier: MQCA_Q_NAME).

The name of the queue to be changed. The maximum length of the string is MQ_Q_NAME_LENGTH.

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

The value specified must match the type of the queue being changed.

The value may be:

MQQT_ALIAS
Alias queue definition.

MQQT_LOCAL
Local queue.

MQQT_REMOTE
Local definition of a remote queue.

Optional parameters

BaseQName (MQCFST)
Queue name to which the alias resolves (parameter identifier: MQCA_BASE_Q_NAME).

This is the name of a local or remote queue that is defined to the local queue manager.

The maximum length of the string is MQ_Q_NAME_LENGTH.

CICSFileName (MQCFST)
CSD file name for queue messages (parameter identifier: MQCA_CICS_FILE_NAME).

This is the name of a VSAM file defined to CICS that is to be associated with a queue definition for storing queue messages.

The maximum length of the string is MQ_CICS_FILE_NAME_LENGTH.

InhibitGet (MQCFIN)
Whether get operations are allowed (parameter identifier: MQIA_INHIBIT_GET).

The value may be:

MQQA_GET_ALLOWED
Get operations are allowed.

MQQA_GET_INHIBITED
Get operations are inhibited.

InhibitPut (MQCFIN)
Whether put operations are allowed (parameter identifier: MQIA_INHIBIT_PUT).

Specifies whether messages can be put on the queue.

The value may be:

MQQA_PUT_ALLOWED
Put operations are allowed.

MQQA_PUT_INHIBITED
Put operations are inhibited.

MaxGlobalLocks (MQCFIN)
Buffer size for queue manager to manage concurrent queue access (parameter identifier: MQIA_MAX_GLOBAL_LOCKS).

The maximum number of entries that the queue manager can use to maintain uncommitted MQPUT or MQGET calls, for each queue in the system, for recovery.

A value of 500 is normally sufficient.

The maximum value is 1000.

MaxLocalLocks (MQCFIN)
Buffer size for applications to manage concurrent queue access (parameter identifier: MQIA_MAX_LOCAL_LOCKS).

The maximum number of entries that the queue manager can use to maintain uncommitted MQPUT or MQGET calls for each queue and task for recovery.

A value of 500 is normally sufficient.

The maximum value is 1000.

MaxMsgLength (MQCFIN)
Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

Specifies the maximum length for messages on the queue. Because applications may use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue, the value should be changed only if it is known that this will not cause an application to operate incorrectly.

You cannot set a value that is greater than the queue manager's MaxMsgLength attribute.

The lower limit for this parameter is 0. The upper limit depends on the environment. For MQSeries for VSE/ESA, the maximum is 4 MB.

MaxQDepth (MQCFIN)
Maximum queue depth (parameter identifier: MQIA_MAX_Q_DEPTH).

The maximum number of messages allowed on the queue. Note that other factors may cause the queue to be treated as full; for example, it will appear to be full if there is no storage available for a message.

You cannot set a value that is greater than the queue manager's MaxQDepth attribute.

Specify a value in the range 0 through 640 000.

MaxQTriggers (MQCFIN)
Maximum number of concurrent trigger instances for a particular queue (parameter identifier: MQIA_MAX_Q_TRIGGERS).

The maximum number of trigger threads that can be activated simultaneously. This parameter applies to queues with a TriggerType of MQTT_EVERY.

Specify a value in the range 1 through 9999.

MaxQUsers (MQCFIN)
Maximum number of active opens to any particular queue (parameter identifier: MQIA_Q_USERS).

The maximum number of active opens requests against a queue.

You cannot set a value that is greater than the queue manager's MaxQUsers attribute.

Specify a value in the range 1 through 32000.

QDepthHighEvent (MQCFIN)
Controls whether Queue Depth High events are generated (parameter identifier: MQIA_Q_DEPTH_HIGH_EVENT).

A Queue Depth High event indicates that an application has put a message on a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the QDepthHighLimit parameter.

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QDepthHighLimit (MQCFIN)
High limit for queue depth (parameter identifier: MQIA_Q_DEPTH_HIGH_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth High event.

This event indicates that an application has put a message to a queue, and this has caused the number of messages on the queue to become greater than or equal to the queue depth high threshold. See the QDepthHighEvent parameter.

The value is expressed as a percentage of the maximum queue depth (MaxQDepth attribute), and must be greater than or equal to zero and less than or equal to 100.

QDepthLowEvent (MQCFIN)
Controls whether Queue Depth Low events are generated (parameter identifier: MQIA_Q_DEPTH_LOW_EVENT).

A Queue Depth Low event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the QDepthLowLimit parameter.

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QDepthLowLimit (MQCFIN)
Low limit for queue depth (parameter identifier: MQIA_Q_DEPTH_LOW_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

This event indicates that an application has retrieved a message from a queue, and this has caused the number of messages on the queue to become less than or equal to the queue depth low threshold. See the QDepthLowEvent parameter.

The value is expressed as a percentage of the maximum queue depth (MaxQDepth attribute), and must be greater than or equal to zero and less than or equal to 100.

QDepthMaxEvent (MQCFIN)
Controls whether Queue Full events are generated (parameter identifier: MQIA_Q_DEPTH_MAX_EVENT).

A Queue Full event indicates that an MQPUT call to a queue has been rejected because the queue is full, that is, the queue depth has already reached its maximum value.

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QDesc (MQCFST)
Queue description (parameter identifier: MQCA_Q_DESC).

Text that briefly describes the object.

The maximum length of the string is MQ_Q_DESC_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the message queue manager on which the command is executing to ensure that the text is translated correctly if it is sent to another queue manager.

QServiceInterval (MQCFIN)
Target for queue service interval (parameter identifier: MQIA_Q_SERVICE_INTERVAL).

The service interval used for comparison to generate Queue Service Interval High and Queue Service Interval OK events. See the QServiceIntervalEvent parameter.

The value is in units of milliseconds, and must be greater than or equal to zero, and less than or equal to 99 999 999.

QServiceIntervalEvent (MQCFIN)
Controls whether Service Interval High or Service Interval OK events are generated (parameter identifier: MQIA_Q_SERVICE_INTERVAL_EVENT).

A Queue Service Interval High event is generated when a check indicates that no messages have been retrieved from or put to the queue for at least the time indicated by the QServiceInterval attribute.

A Queue Service Interval OK event is generated when a check indicates that a message has been retrieved from the queue within the time indicated by the QServiceInterval attribute.

Note: The value of this attribute can change implicitly. See "Performance events" on page 75.

The value may be:

- MQQSIE_HIGH Queue Service Interval High events enabled.
  - Queue Service Interval High events are enabled and
  - Queue Service Interval OK events are disabled.
- MQQSIE_OK Queue Service Interval OK events enabled.
  - Queue Service Interval High events are disabled and
  - Queue Service Interval OK events are enabled.
- MQQSIE_NONE No queue service interval events enabled.
  - Queue Service Interval High events are disabled and
  - Queue Service Interval OK events are also disabled.

RemoteQName (MQCFST)
Name of remote queue as known locally on the remote queue manager (parameter identifier: MQCA_REMOTE_Q_NAME).

If this definition is used for a local definition of a remote queue, RemoteQName must not be blank when the open occurs.

If this definition is used for a queue-manager alias definition, RemoteQName must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The maximum length of the string is MQ_Q_NAME_LENGTH.

RemoteQMgrName (MQCFST)
Name of remote queue manager (parameter identifier: MQCA_REMOTE_Q_MGR_NAME).

If an application opens the local definition of a remote queue, RemoteQMgrName must not be blank or the name of the connected queue manager. If XmitQName is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue-manager alias, RemoteQMgrName is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if XmitQName is blank, when the queue is opened there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

Shareability (MQCFIN)
Whether queue can be shared (parameter identifier: MQIA_SHAREABILITY).

Specifies whether multiple instances of applications, can open this queue for input.

The value may be:

MQQA_SHAREABLE
Queue is shareable.

MQQA_NOT_SHAREABLE
Queue is not shareable.

TriggerChannelName (MQCFST)
Channel name for MCA trigger process (parameter identifier: MQCA_TRIGGER_CHANNEL_NAME).

This parameter is only valid for queues with Usage of MQUS_TRANSMISSION.

For a transmission queue definition, this parameter must identify a channel name.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

TriggerControl (MQCFIN)
Trigger control (parameter identifier: MQIA_TRIGGER_CONTROL).

Specifies whether message activity on a queue will cause a trigger transaction or program to be started.

The value may be:

MQTC_OFF
Trigger activation not required.

MQTC_ON
Trigger activation required.

TriggerData (MQCFST)
Trigger data (parameter identifier: MQCA_TRIGGER_DATA).

Specifies user data that the queue manager includes in the trigger data structure that is passed to a triggered transaction or program.

The maximum length of the string is MQ_PROCESS_USER_DATA_LENGTH.

TriggerProgramName (MQCFST)
Program name for trigger process (parameter identifier: MQCA_TRIGGER_PROGRAM_NAME).

Specifies the program name that is to be started when a trigger event occurs on the queue.

The maximum length of the string is MQ_TRIGGER_PROGRAM_NAME_LENGTH.

TriggerRestart (MQCFIN)
Indicator for the reactivation of a trigger process (parameter identifier: MQIA_TRIGGER_RESTART).

Specifies whether or not a trigger instance can be restarted when it is detected that there are messages on a queue, but no trigger instance already running.

The value may be:

MQTRIGGER_RESTART_NO
Trigger reactivation not required.

MQTRIGGER_RESTART_YES
Trigger reactivation required.

TriggerTerminalId (MQCFST)
Terminal identifier for trigger process (parameter identifier: MQCA_TRIGGER_TERM_ID).

Specifies a CICS terminal identifier to be associated with a trigger transaction or program instance.

The maximum length of the string is MQ_TRIGGER_TERM_ID_LENGTH.

TriggerTransactionId (MQCFST)
Transaction identifier for trigger process (parameter identifier: MQCA_TRIGGER_TRANS_ID).

Specifies a transaction identifier is to be started when a trigger event occurs on the queue.

The maximum length of the string is MQ_TRIGGER_TRANS_ID_LENGTH.

TriggerType (MQCFIN)
Trigger type (parameter identifier: MQIA_TRIGGER_TYPE).

Specifies the condition that initiates a trigger event. When the condition is true, a trigger transaction or program is started.

The value may be:

MQTT_NONE
No trigger activation.

MQTT_EVERY
Trigger activation for every message.

MQTT_FIRST
Trigger activation when queue depth goes from 0 to 1.

Usage (MQCFIN)
Usage (parameter identifier: MQIA_USAGE).

Specifies whether the queue is for normal usage or for transmitting messages to a remote message queue manager.

The value may be:

MQUS_NORMAL
Normal usage.

MQUS_TRANSMISSION
Transmission queue.

XmitQName (MQCFST)
Transmission queue name (parameter identifier: MQCA_XMIT_Q_NAME).

Specifies the local name of the transmission queue to be used for messages destined for either a remote queue or for a queue-manager alias definition.

If XmitQName is blank, a queue with the same name as RemoteQMgrName is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue-manager alias and RemoteQMgrName is the name of the connected queue manager.

It is also ignored if the definition is used as a reply-to queue alias definition.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Note that the optional parameters described in this section are also applicable to these PCF commands:

- "Copy Queue" on page 200
- "Create Queue" on page 204

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.

MQRCCF_ATTR_VALUE_ERROR
Attribute value not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_OBJECT_NAME_ERROR
Object name not valid.

MQRCCF_OBJECT_OPEN
Object is open.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Change Queue Manager

The Change Queue Manager (MQCMD_CHANGE_Q_MGR) command changes the specified attributes of the queue manager.

This PCF is supported on all platforms.

For any optional parameters that are omitted, the value does not change.

Required parameters:

None

Optional parameters:

AuthorityEvent, BatchInterfaceAutoStart, BatchInterfaceId, CodedCharSetId, CommandInputQName, CommandReplyQName, CommandServerAutoStart, CommandServerDataConversion, CommandServerDeadLetterQ, DeadLetterQName, InhibitEvent, ListenerPortNumber, LocalEvent, MaxGlobalLocks, MaxHandles, MaxLocalLocks, MaxMsgLength, MaxOpenQ, MaxQDepth, MaxQUsers, MonitorInterval, MonitorQName, PerformanceEvent, QMgrDesc, RemoteEvent, SSLKeyLibraryMember, SSLKeyLibraryName, StartStopEvent, SystemLogQName

Optional parameters

AuthorityEvent (MQCFIN)
Controls whether authorization (Not Authorized) events are generated (parameter identifier: MQIA_AUTHORITY_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

BatchInterfaceAutoStart (MQCFIN)
Indicator for the automatic activation of the batch interface (parameter identifier: MQIA_BATCH_INTERFACE_AUTO).

Specifies whether the MQSeries for VSE/ESA batch interface is automatically started when the the local queue manager is started.

The value may be:

MQAUTO_START_NO
Do not auto-start the batch interface.

MQAUTO_START_YES
Auto-start the batch interface.

BatchInterfaceId (MQCFST)
Batch interface identifier (parameter identifier: MQCA_BATCH_INTERFACE_ID).

Specifies a unique batch interface identifier. Batch programs should specify a SETPARM card in their JCL that identifies the appropriate batch interface identifier. For VSE/ESA systems running multiple queue managers, the identifier must be unique.

The maximum length of the string is MQ_BATCH_INTERFACE_ID_LENGTH.

CodedCharSetId (MQCFIN)
Queue manager coded character set identifier (parameter identifier: MQIA_CODED_CHAR_SET_ID).

The coded character set identifier (CCSID) for the queue manager. The CCSID is the identifier used with all character string fields defined by the application programming interface (API). It does not apply to application data carried in the text of a message unless the CCSID in the message descriptor, when the message is put with an MQPUT or MQPUT1, is set to the value MQCCSI_Q_MGR.

Specify a value in the range 1 through 65 535.

The CCSID must specify a value that is defined for use on the platform and use an appropriate character set. For a list of supported CCSIDs for MQSeries for VSE/ESA, review your LE/VSE configuration.

CommandInputQName (MQCFST)
PCF command input queue (parameter identifier: MQCA_COMMAND_INPUT_Q_NAME).

Specifies the name of the PCF command input queue. The PCF command input queue expects PCF messages from local or remote administration applications, and is monitored by the MQSeries Command Server. If this parameter is changed, you are advised to stop and restart the command server so that a new queue name is recognised.

The maximum length of the string is MQ_Q_NAME_LENGTH.

CommandReplyQName (MQCFST)
MQSeries command reply queue (parameter identifier: MQCA_COMMAND_REPLY_Q_NAME).

Specifies the name of the MQSeries command reply queue. The MQSeries command reply queue is used by the MQSeries command utility for responses to MQSeries commands issued from a batch partition.

The maximum length of the string is MQ_Q_NAME_LENGTH.

CommandServerAutoStart (MQCFIN)
Indicator for the automatic activation of the PCF command server (parameter identifier: MQIA_CMD_SERVER_AUTO).

Specifies whether the MQSeries for VSE/ESA PCF command server is automatically started when the the local queue manager is started.

The value may be:

MQAUTO_START_NO
Do not auto-start the PCF command server.

MQAUTO_START_YES
Auto-start the PCF command server.

CommandServerDataConversion (MQCFIN)
Indicator for the data conversion of PCF messages (parameter identifier: MQIA_CMD_SERVER_CONVERT_MSG).

Specifies whether the MQSeries for VSE/ESA PCF command server is to apply data conversion to PCF messages read from the PCF command input queue.

The value may be:

MQCSRV_CONVERT_NO
Do not convert PCF messages.

MQCSRV_CONVERT_YES
Convert PCF messages.

CommandServerDeadLetterQ (MQCFST)
Indicator for the storage of undeliverable PCF reply messages to the system dead letter queue(parameter identifier: MQIA_CMD_SERVER_DLQ_MSG).

Specifies whether the MQSeries for VSE/ESA PCF command server is to place undeliverable PCF command responses to the system dead letter queue.

The value may be:

MQCSRV_DLQ_NO
Do not put undeliverable PCF responses to the system dead letter queue.

MQCSRV_DLQ_YES
Put undeliverable PCF responses to the system dead letter queue.

DeadLetterQName (MQCFST)
Dead letter (undelivered message) queue name (parameter identifier: MQCA_DEAD_LETTER_Q_NAME).

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The maximum length of the string is MQ_Q_NAME_LENGTH.

InhibitEvent (MQCFIN)
Controls whether inhibit (Inhibit Get and Inhibit Put) events are generated (parameter identifier: MQIA_INHIBIT_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

ListenerPortNumber (MQCFIN)
Port number for TCP/IP Listener process (parameter identifier: MQIA_LISTENER_PORT_NUMBER).

Specifies the TCP/IP port number that MQSeries uses for accepting TCP/IP connection requests from remote queue managers and MQ clients. The default port value is 1414, however any unreserved port number can be used.

LocalEvent (MQCFIN)
Controls whether local error events are generated (parameter identifier: MQIA_LOCAL_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

MaxGlobalLocks (MQCFIN)
Buffer size for queue manager to manage concurrent queue access (parameter identifier: MQIA_MAX_GLOBAL_LOCKS).

The maximum number of entries that the queue manager can use to maintain uncommitted MQPUT or MQGET calls, for each queue in the system, for recovery.

A value of 500 is normally sufficient.

The maximum value is 1000.

MaxHandles (MQCFIN)
Maximum number of handles (parameter identifier: MQIA_MAX_HANDLES).

The maximum number of connection handles that the queue manager will manage at any one time.

Specify a value in the range 1 through 1000.

MaxLocalLocks (MQCFIN)
Buffer size for applications to manage concurrent queue access (parameter identifier: MQIA_MAX_LOCAL_LOCKS).

The maximum number of entries that the queue manager can use to maintain uncommitted MQPUT or MQGET calls for each queue and task for recovery.

A value of 500 is normally sufficient.

The maximum value is 1000.

MaxMsgLength (MQCFIN)
Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

Specifies the maximum length of messages allowed on queues on the queue manager. Changing this parameter does not affect existing queue definitions.

If you reduce the maximum message length for the queue manager, you should verify that existing queues do not already exceed the new MaxMsgLength value.

The lower limit for this parameter is 0. The upper limit for MQSeries for VSE/ESA is 4 MB.

MaxQOpen (MQCFIN)
Maximum number of concurrently open queues (parameter identifier: MQIA_MAX_OPEN_Q).

Specifies the maximum number of queues any single task can have open at any one time.

Specify a value in the range 1 through 1000.

MaxQDepth (MQCFIN)
Maximum queue depth (parameter identifier: MQIA_MAX_Q_DEPTH).

The maximum number of messages allowed on a queue. Note that other factors may cause the queue to be treated as full; for example, it will appear to be full if there is no storage available for a message.

If you reduce the maximum queue depth for the queue manager, you should verify that existing queues do not already exceed the new MaxQDepth value.

Specify a value in the range 0 through 640 000.

MaxQUsers (MQCFIN)
Maximum number of active opens to any particular queue (parameter identifier: MQIA_Q_USERS).

The maximum number of open requests that the queue manager will manage for a single queue.

Specify a value in the range 1 through 32000.

MonitorInterval (MQCFIN)
Queue manager housekeeping process interval (parameter identifier: MQIA_MONITOR_INTERVAL).

Specifies the interval (in seconds) that the queue manager housekeeping task suspends during process iterations.

A value of 30 seconds is usually sufficient.

MonitorQName (MQCFST)
MQI monitor queue name (parameter identifier: MQCA_MONITOR_Q_NAME).

Specifies the name of the local queue that is to be used for MQI diagnostic messages. The MQI Monitor when active, places diagnostic messages to the monitor queue.

The maximum length of the string is MQ_Q_NAME_LENGTH.

PerformanceEvent (MQCFIN))
Controls whether performance-related events are generated (parameter identifier: MQIA_PERFORMANCE_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QMgrDesc (MQCFST)
Queue manager description (parameter identifier: MQCA_Q_MGR_DESC).

This is text that briefly describes the object.

The maximum length of the string is MQ_Q_MGR_DESC_LENGTH.

Use characters from the character set identified by the coded character set identifier (CCSID) for the queue manager on which the command is executing, to ensure that the text is translated correctly.

RemoteEvent (MQCFIN)
Controls whether remote error events are generated (parameter identifier: MQIA_REMOTE_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

SSLKeyLibraryMember (MQCFST)
SSL key library name (parameter identifier: MQCA_SSL_KEY_LIBRARY).

Specifies the SSL key-ring sublibrary. The key-ring sublibrary contains private key and X.509 certificate files.

Specify a valid VSE sublibrary name.

The maximum length of the string is MQ_SSL_KEY_LIBRARY_LENGTH.

SSLKeyLibraryName (MQCFST)
SSL key member name (parameter identifier: MQCA_SSL_KEY_MEMBER).

Specifies the SSL key-ring member name of the private key and certificate files that will be used by MQSeries enabled channels.

This must be a valid VSE sublibrary member name.

It should be noted that MQSeries for VSE/ESA uses the same private key and certificate for all SSL enabled channels. It is not possible to identify a different certificate on a per channel basis. Consequently, the key-ring member name should identify a private key and certificate files appropriate for all SSL enabled channels.

The maximum length of the string is MQ_SSL_KEY_MEMBER_LENGTH.

StartStopEvent (MQCFIN)
Controls whether start and stop events are generated (parameter identifier: MQIA_START_STOP_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

SystemLogQName (MQCFST)
System log queue name (parameter identifier: MQCA_SYSTEM_LOG_Q_NAME).

Specifies the name of the system log queue that is used by MQSeries for VSE/ESA to store operational diagnostic and error messages.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_ATTR_VALUE_ERROR
Attribute value not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_OBJECT_NAME_ERROR
Object name not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_Q_MGR_CCSID_ERROR
Coded character set value not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

MQRCCF_UNKNOWN_Q_MGR
Queue manager not known.

Copy Channel

The Copy Channel (MQCMD_COPY_CHANNEL) command creates a new channel definition using, for attributes not specified in the command, the attribute values of an existing channel definition.

This PCF is supported on all platforms.

Required parameters:

FromChannelName, ToChannelName, ChannelType

Optional parameters:

AllocRetryCount, AllocRetryFastTimer, AllocRetrySlowTimer, BatchSize, ChannelDesc, ConnectionName, DataConversion, DiscInterval, DiscRetryCount, MaxMsgLength, MsgExit, MsgUserData, PortNumber, ReceiveExit, ReceiveUserData, SecurityExit, SecurityUserData, SendExit, SendUserData, SeqNumberWrap, SSLCipherSpec, SSLClientAuth, SSLPeerName, TpName, TransportType, XmitQName

Required parameters

FromChannelName (MQCFST)
From channel name (parameter identifier: MQCACF_FROM_CHANNEL_NAME).

The name of the existing channel definition that contains values for the attributes that are not specified in this command.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ToChannelName (MQCFST)
To channel name (parameter identifier: MQCACF_TO_CHANNEL_NAME).

The name of the new channel definition.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Channel names must be unique; if a channel definition with this name already exists, the command will fail.

ChannelType (MQCFIN)
Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

Specifies the type of the channel being copied. The value may be:

MQCHT_SENDER
Sender.

MQCHT_RECEIVER
Receiver.

MQCHT_SVRCONN
Server-connection (for use by clients).

Optional parameters

For a complete list and description of the optional parameters available with the Copy Channel command, refer to "Change channel" on page 176.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_BATCH_SIZE_ERROR
Batch size not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFSL_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFSL_TOTAL_LENGTH_ERROR
Total string length error.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_ALREADY_EXISTS
Channel already exists.

MQRCCF_CHANNEL_NAME_ERROR
Channel name error.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_CONN_NAME_ERROR
Error in connection name parameter.

MQRCCF_DISC_INT_ERROR
Disconnection interval not valid.

MQRCCF_MAX_MSG_LENGTH_ERROR
Maximum message length not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_SEQ_NUMBER_WRAP_ERROR
Sequence wrap number not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

MQRCCF_XMIT_PROTOCOL_TYPE_ERR
Transmission protocol type not valid.

MQRCCF_XMIT_Q_NAME_ERROR
Transmission queue name error.

MQRCCF_XMIT_Q_NAME_WRONG_TYPE
Transmission queue name not allowed for this channel type.

Copy Queue

The Copy Queue (MQCMD_COPY_Q) command creates a new queue definition, of the same type, using, for attributes not specified in the command, the attribute values of an existing queue definition.

This PCF is supported on all platforms.

Required parameters:

FromQName, ToQName, QType

Optional parameters:

BaseQName, CICSFileName, InhibitGet, InhibitPut, MaxGlobalLocks, MaxLocalLocks, MaxMsgLength, MaxQDepth, MaxQTriggers, MaxQUsers, QDepthHighEvent, QDepthHighLimit, QDepthLowEvent, QDepthLowLimit, QDepthMaxEvent, QDesc, QServiceInterval, QServiceIntervalEvent, RemoteQMgrName, RemoteQName, Shareability, TriggerChannelName, TriggerControl, TriggerData, TriggerProgramName, TriggerRestart, TriggerTerminalId, TriggerTransactionId, TriggerType, Usage, XmitQName

Required parameters

FromQName (MQCFST)
From queue name (parameter identifier: MQCACF_FROM_Q_NAME).

Specifies the name of the existing queue definition.

The maximum length of the string is MQ_Q_NAME_LENGTH.

ToQName (MQCFST)
To queue name (parameter identifier: MQCACF_TO_Q_NAME).

Specifies the name of the new queue definition.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Queue names must be unique; if a queue definition exists with the same name as the new queue, the command will fail.

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

The value specified must match the type of the queue being copied.

The value may be:

MQQT_ALIAS
Alias queue definition.

MQQT_LOCAL
Local queue.

MQQT_REMOTE
Local definition of a remote queue.

Optional parameters

For a complete list and description of the optional parameters available with the Copy Queue command, refer to “Change queue” on page 183.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.

MQRCCF_ATTR_VALUE_ERROR
Attribute value not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_LIKE_OBJECT_WRONG_TYPE
New and existing objects have different type.

MQRCCF_OBJECT_ALREADY_EXISTS
Object already exists.

MQRCCF_OBJECT_NAME_ERROR
Object name not valid.

MQRCCF_OBJECT_OPEN
Object is open.

MQRCCF_OBJECT_WRONG_TYPE
Object has wrong type.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Create Channel

The Create Channel (MQCMD_CREATE_CHANNEL) command creates an MQSeries channel definition. Any attributes that are not defined explicitly are set to the default values on the destination queue manager.

This PCF is supported on all platforms.

Required parameters:

ChannelName, ChannelType

Optional parameters:

AllocRetryCount, AllocRetryFastTimer, AllocRetrySlowTimer, BatchSize, ChannelDesc, ConnectionName, DataConversion, DiscInterval, DiscRetryCount, MaxMsgLength, MsgExit, MsgUserData, PortNumber, ReceiveExit, ReceiveUserData, SecurityExit, SecurityUserData, SendExit, SendUserData, SeqNumberWrap, SSLCipherSpec, SSLClientAuth, SSLPeerName, TpName, TransportType, XmitQName

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the new channel definition. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Channel names must be unique; if a channel definition with this name already exists, the command will fail.

ChannelType (MQCFIN)

Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

Specifies the type of the channel being defined. The value may be:

MQCHT_SENDER
Sender.

MQCHT_RECEIVER
Receiver.

MQCHT_SVRCONN
Server-connection (for use by clients).

Optional parameters

For a complete list and description of the optional parameters available with the Copy Channel command, refer to “Change channel” on page 176.

Error codes

In addition to the values for any command shown in section “Error codes applicable to all commands” on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_BATCH_SIZE_ERROR
Batch size not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFSL_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFSL_TOTAL_LENGTH_ERROR
Total string length error.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_ALREADY_EXISTS
Channel already exists.

MQRCCF_CHANNEL_NAME_ERROR
Channel name error.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_CONN_NAME_ERROR
Error in connection name parameter.

MQRCCF_DISC_INT_ERROR
Disconnection interval not valid.

MQRCCF_MAX_MSG_LENGTH_ERROR
Maximum message length not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_SEQ_NUMBER_WRAP_ERROR
Sequence wrap number not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

MQRCCF_XMIT_PROTOCOL_TYPE_ERR
Transmission protocol type not valid.

MQRCCF_XMIT_Q_NAME_ERROR
Transmission queue name error.

Create Queue

The Create Queue (MQCMD_CREATE_Q) command creates a queue definition with the specified attributes. All attributes that are not specified are set to the default value for the type of queue that is created.

This PCF is supported on all platforms.

Required parameters:

QName, QType, CICSFileName

Optional parameters:

BaseQName, InhibitGet, InhibitPut, MaxGlobalLocks, MaxLocalLocks, MaxMsgLength, MaxQDepth, MaxQTriggers, MaxQUsers, QDepthHighEvent, QDepthHighLimit, QDepthLowEvent, QDepthLowLimit, QDepthMaxEvent, QDesc, QServiceInterval, QServiceIntervalEvent, RemoteQMgrName, RemoteQName, Shareability, TriggerChannelName, TriggerControl, TriggerData, TriggerProgramName, TriggerRestart, TriggerTerminalId, TriggerTransactionId, TriggerType, Usage, XmitQName

Required parameters

QName (MQCFST)
Queue name (parameter identifier: MQCA_Q_NAME).

The name of the queue to be created. The maximum length of the string is MQ_Q_NAME_LENGTH.

Queue names must be unique; if a queue definition already exists with the same name as of the new queue, the command will fail.

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

The value may be:

MQQT_ALIAS
Alias queue definition.

MQQT_LOCAL
Local queue.

MQQT_REMOTE
Local definition of a remote queue.

CICSFileName (MQCFST)
CICS file name for queue messages.

The name of a filename defined to the CICS region. The maximum length of the string is MQ_CICS_FILE_NAME_LENGTH.

Optional parameters

For a complete list and description of the optional parameters available with the Create Queue command, refer to “Change queue” on page 183.

Error codes

In addition to the values for any command shown in section “Error codes applicable to all commands” on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.

MQRCCF_ATTR_VALUE_ERROR
Attribute value not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_OBJECT_ALREADY_EXISTS
Object already exists.

MQRCCF_OBJECT_NAME_ERROR
Object name not valid.

MQRCCF_OBJECT_OPEN
Object is open.

MQRCCF_OBJECT_WRONG_TYPE
Object has wrong type.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Delete Channel

The Delete Channel (MQCMD_DELETE_CHANNEL) command deletes the specified channel definition.

This PCF is supported on all platforms.

Required parameters:

ChannelName

Optional parameters:

None

Required parameters

ChannelName (MQCFST)

Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel definition to be deleted. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Delete Queue

The Delete Queue (MQCMD_DELETE_Q) command deletes an MQSeries queue.

This PCF is supported on all platforms.

Required parameters:

QName

Optional parameters:

Purge, QType

Required parameters

QName (MQCFST)

Queue name (parameter identifier: MQCA_Q_NAME).

The name of the queue to be deleted.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

If this parameter is present, the queue must be of the specified type.

The value may be:

MQQT_ALIAS
Alias queue definition.

MQQT_LOCAL
Local queue.

MQQT_REMOTE
Local definition of a remote queue.

Purge (MQCFIN)
Purge queue (parameter identifier: MQIACF_PURGE).

If there are messages on the queue MQPO_YES must be specified, otherwise the command will fail. If this parameter is not present the queue is not purged.

Valid only for queue of type local.

The value may be:

MQPO_YES
Purge the queue.

MQPO_NO
Do not purge the queue.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_Q_NOT_EMPTY
(2055, X'807') Queue contains one or more messages or uncommitted put or get requests.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_OBJECT_OPEN
Object is open.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PURGE_VALUE_ERROR
Purge value not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Escape

The Escape (MQCMD_ESCAPE) command conveys any MQSeries command to a remote queue manager.

The Escape command can also be used to send a command for which no Programmable Command Format has been defined.

The only type of command that can be carried is one that is identified as an MQSeries command that is recognized at the receiving queue manager.

Required parameters:
EscapeType, EscapeText

Optional parameters:
None

The Escape command, if successful, generates a data response. For details of the Escape response, refer to "Data responses to commands" on page 226.

Required parameters

EscapeType (MQCFIN)
Escape type (parameter identifier: MQIACF_ESCAPE_TYPE).

The only value supported is:

MQET_MQSC
MQSeries command.

EscapeText (MQCFST)
Escape text (parameter identifier: MQCACF_ESCAPE_TEXT).

A string to hold a command. The length of the string is limited only by the size of the message. MQSeries for VSE/ESA supports PCF message lengths up to 2 KB.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_ESCAPE_TYPE_ERROR
Escape type not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_PARM_SEQUENCE_ERROR
Parameter sequence not valid.

Inquire Channel

The Inquire Channel (MQCMD_INQUIRE_CHANNEL) command inquires about the attributes of MQSeries channel definitions.

This PCF is supported on all platforms.

Required parameters:

ChannelName

Optional parameters:

ChannelType, ChannelAttrs

The Inquire Channel command, if successful, generates a data response. For details of the Inquire Channel response, refer to "Data responses to commands" on page 226.

Required parameters

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Generic channel names are not supported on MQSeries for VSE/ESA.

The channel name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelType (MQCFIN)
Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

If this parameter is present, the channel specified by ChannelName, must be of the specified type.

If this parameter is not present (or if MQCHT_ALL is specified), the channel identified by ChannelName can be of any type.

The value may be:

MQCHT_SENDER
Sender.

MQCHT_RECEIVER
Receiver.

MQCHT_SVRCONN
Server-connection (for use by clients).

MQCHT_ALL
All types.

The default value if this parameter is not specified is MQCHT_ALL. Note: If this parameter is present, it must occur immediately after the ChannelName parameter. Failure to do this can result in a MQRCCF_MSG_LENGTH_ERROR error message.

ChannelAttrs (MQCFIL)
Channel attributes (parameter identifier: MQIACF_CHANNEL_ATTRS).

The attribute list may specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL
All attributes.

or a combination of:

MQCA_ALTERATION_DATE
Date on which the definition was last altered.

MQCA_ALTERATION_TIME
Time at which the definition was last altered.

MQIACH_ALLOC_RETRY
APPC connection retry count.

MQIACH_ALLOC_FAST_TIMER
APPC connection retry fast timer.

MQIACH_ALLOC_SLOW_TIMER
APPC connection retry slow timer.

MQIACH_BATCH_SIZE
Batch size.

MQCACH_DESC
Description.

MQCACH_CHANNEL_NAME
Channel name.

MQIACH_CHANNEL_TYPE
Channel type.

MQCACH_CONNECTION_NAME
Connection name.

MQIACH_DATA_CONVERSION
Whether sender should convert application data.

MQIACH_DISC_INTERVAL
Disconnection interval.

MQIACH_DISC_RETRY
Disconnection retry count.

MQIACH_MAX_MSG_LENGTH
Maximum message length.

MQCACH_MSG_EXIT_NAME
Message exit name.

MQCACH_MSG_EXIT_USER_DATA
Message exit user data.

MQCACH_RCV_EXIT_NAME
Receive exit name.

MQCACH_RCV_EXIT_USER_DATA
Receive exit user data.

MQCACH_SEC_EXIT_NAME
Security exit name.

MQCACH_SEC_EXIT_USER_DATA
Security exit user data.

MQCACH_SEND_EXIT_NAME
Send exit name.

MQCACH_SEND_EXIT_USER_DATA
Send exit user data.

MQIACH_PORT_NUMBER
TCP/IP port number.

MQIACH_SEQUENCE_NUMBER_WRAP
Sequence number wrap.

MQCACH_SSL_CIPHER_SPEC
SSL cipher specification.

MQIACH_SSL_CLIENT_AUTH
SSL client authentication.

MQCACH_SSL_PEER_NAME
SSL peer name.

MQCACH_TP_NAME
Transaction program name.

MQIACH_XMIT_PROTOCOL_TYPE
Transport (transmission protocol) type.

MQCACH_XMIT_Q_NAME
Transmission queue name.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_SELECTOR_ERROR
(2067, X'813') Attribute selector not valid.

MQRCCF_CFIL_COUNT_ERROR
Count of parameter values not valid.

MQRCCF_CFIL_DUPLICATE_VALUE
Duplicate parameter.

MQRCCF_CFIL_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIL_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_NAME_ERROR
Channel name error.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Inquire Channel Names

The Inquire Channel Names (MQCMD_INQUIRE_CHANNEL_NAMES) command inquires a list of MQSeries channel names that match the generic channel name, and the optional channel type specified.

This PCF is supported on all platforms.

Required parameters:
ChannelName

Optional parameters:
ChannelType

The Inquire Channel Names command, if successful, generates a data response. For details of the Inquire Channel Names response, refer to “Data responses to commands” on page 226.

Required parameters

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

Generic channel names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

ChannelType (MQCFIN)
Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

If present, this parameter limits the channel names returned to channels of the specified type.

The value may be:

MQCHT_SENDER
Sender.

MQCHT_RECEIVER
Receiver.

MQCHT_SVRCONN
Server-connection (for use by clients).

MQCHT_ALL
All types.

The default value if this parameter is not specified is MQCHT_ALL, which means that channels of all types are eligible.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_NAME_ERROR
Channel name error.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Inquire Queue

The Inquire Queue (MQCMD_INQUIRE_Q) command inquires about the attributes of MQSeries queues.

This PCF is supported on all platforms.

Required parameters:

QName

Optional parameters:

QType, QAttrs

The Inquire Queue command, if successful, generates a data response. For details of the Inquire Queue response, refer to "Data responses to commands" on page 226.

Required parameters

QName (MQCFST)
Queue name (parameter identifier: MQCA_Q_NAME).

Generic queue names are not supported by MQSeries for VSE/ESA.

The queue name is always returned, regardless of the attributes requested.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

If this parameter is present, the queue identified by QName must be of the specified type.

If this parameter is not present (or if MQQT_ALL is specified), the queue identified by QName can be of any type.

The value may be:

MQQT_ALL
All queue types.

MQQT_LOCAL
Local queue.

MQQT_ALIAS
Alias queue definition.

MQQT_REMOTE
Local definition of a remote queue.

QAttrs (MQCFIL)
Queue attributes (parameter identifier: MQIACF_Q_ATTRS).

The attribute list may specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL
All attributes.

or a combination of:

MQCA_ALTERATION_DATE
The date on which the information was last altered, in the form yyyy-mm-dd.

MQCA_ALTERATION_TIME
The time at which the information was last altered, in the form hh.mm.ss.

MQCA_BASE_Q_NAME
Name of queue that alias resolves to.

MQCA_CICS_FILE_NAME
CSD file name for queue messages.

MQCA_CREATION_DATE
Queue creation date.

MQCA_CREATION_TIME
Queue creation time.

MQIA_INHIBIT_GET
Whether get operations are allowed.

MQIA_INHIBIT_PUT
Whether put operations are allowed.

MQIA_MAX_GLOBAL_LOCKS
Buffer size for queue manager to manage concurrent queue access.

MQIA_MAX_LOCAL_LOCKS
Buffer size for applications to manage concurrent queue access.

MQIA_MAX_MSG_LENGTH
Maximum message length.

MQIA_MAX_Q_DEPTH
Maximum number of messages allowed on queue.

MQIA_MAX_Q_TRIGGERS
Maximum number of concurrent trigger instances for a particular queue.

MQIA_Q_DEPTH_HIGH_EVENT
Control attribute for queue depth high events.

MQIA_Q_DEPTH_HIGH_LIMIT
High limit for queue depth.

MQIA_Q_DEPTH_LOW_EVENT
Control attribute for queue depth low events.

MQIA_Q_DEPTH_LOW_LIMIT
Low limit for queue depth.

MQIA_Q_DEPTH_MAX_EVENT
Control attribute for queue depth max events.

MQCA_Q_NAME
Queue name.

MQIA_Q_SERVICE_INTERVAL
Limit for queue service interval.

MQIA_Q_SERVICE_INTERVAL_EVENT
Control attribute for queue service interval events.

MQIA_Q_TYPE
Queue type.

MQCA_Q_DESC
Queue description.

MQIA_Q_USERS
Maximum number of active opens to any particular queue.

MQCA_REMOTE_Q_NAME
Name of remote queue as known locally on the remote queue manager.

MQCA_REMOTE_Q_MGR_NAME
Name of remote queue manager.

MQIA_SHAREABILITY
Whether queue can be shared.

MQCA_TRIGGER_CHANNEL_NAME
Channel name for MCA trigger process.

MQIA_TRIGGER_CONTROL
Trigger control.

MQCA_TRIGGER_DATA
Trigger data.

MQCA_TRIGGER_PROGRAM_NAME
Program name for trigger process.

MQIA_TRIGGER_RESTART
Indicator for the reactivation of a trigger process.

MQCA_TRIGGER_TERM_ID
Terminal identifier for trigger process.

MQCA_TRIGGER_TRANS_ID
Transaction identifier for trigger process.

MQIA_TRIGGER_TYPE
Trigger type.

MQIA_USAGE
Usage.

MQCA_XMIT_Q_NAME
Transmission queue name.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_SELECTOR_ERROR
(2067, X'813') Attribute selector not valid.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.

MQRCCF_CFIL_COUNT_ERROR
Count of parameter values not valid.

MQRCCF_CFIL_DUPLICATE_VALUE
Duplicate parameter.

MQRCCF_CFIL_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIL_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Inquire Queue Manager

The Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) command inquires about the attributes of a queue manager.

This PCF is supported on all platforms.

Required parameters:

None

Optional parameters:

QMgrAttrs

The Inquire Queue Manager command, if successful, generates a data response. For details of the Inquire Queue Manager response, refer to "Data responses to commands" on page 226.

Optional parameters

QMgrAttrs (MQCFIL)
Queue manager attributes (parameter identifier: MQIACF_Q_MGR_ATTRS).

The attribute list may specify the following on its own (this is the default value used if the parameter is not specified):

MQIACF_ALL
All attributes.

or a combination of:

MQCA_ALTERATION_DATE
Date at which the definition was last altered.

MQCA_ALTERATION_TIME
Time at which the definition was last altered.

MQIA_AUTHORITY_EVENT
Control attribute for authority events.

MQIA_BATCH_INTERFACE_AUTO
Indicator for the automatic activation of the batch interface.

MQCA_BATCH_INTERFACE_ID
Batch interface identifier.

MQIA_CODED_CHAR_SET_ID
Coded character set identifier.

MQCA_COMMAND_INPUT_Q_NAME
System command input queue name.

MQCA_COMMAND_REPLY_Q_NAME
MQSeries command reply queue.

MQIA_CMD_SERVER_AUTO
Indicator for the automatic activation of the PCF command server.

MQIA_CMD_SERVER_CONVERT_MSG
Indicator for the data conversion of PCF messages.

MQIA_CMD_SERVER_DLQ_MSG
Indicator for the storage of undeliverable PCF reply messages to the system dead letter queue.

MQIA_COMMAND_LEVEL
Command level supported by queue manager.

MQCA_DEAD_LETTER_Q_NAME
Name of dead-letter queue.

MQIA_DIST_LISTS
Distribution list support.

MQIA_INHIBIT_EVENT
Control attribute for inhibit events.

MQIA_LISTENER_PORT_NUMBER
Port number for TCP/IP Listener process.

MQIA_LOCAL_EVENT
Control attribute for local events.

MQIA_MAX_GLOBAL_LOCKS
Buffer size for queue manager to manage concurrent queue access.

MQIA_MAX_HANDLES
Maximum number of handles.

MQIA_MAX_LOCAL_LOCKS
Buffer size for applications to manage concurrent queue access.

MQIA_MAX_MSG_LENGTH
Maximum message length.

MQIA_MAX_OPEN_Q
Maximum number of concurrently open queues

MQIA_MAX_Q_DEPTH
Maximum queue depth.

MQIA_PLATFORM
Platform on which the queue manager resides.

MQCA_Q_MGR_DESC
Queue manager description.

MQCA_Q_MGR_NAME
Name of local queue manager.

MQIA_Q_USERS
Maximum number of active opens to any particular queue.

MQIA_REMOTE_EVENT
Control attribute for remote events.

MQIA_START_STOP_EVENT
Control attribute for start stop events.

MQIA_PERFORMANCE_EVENT
Control attribute for performance events.

MQIA_MONITOR_INTERVAL
Queue manager housekeeping process interval.

MQCA_MONITOR_Q_NAME
MQI monitor queue name.

MQCA_SSL_KEY_LIBRARY
SSL key library name.

MQCA_SSL_KEY_MEMBER
SSL key member name.

MQIA_SYNCPOINT
Syncpoint availability.

MQCA_SYSTEM_LOG_Q_NAME
System log queue name.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRC_SELECTOR_ERROR
(2067, X'813') Attribute selector not valid.

MQRCCF_CFIL_COUNT_ERROR
Count of parameter values not valid.

MQRCCF_CFIL_DUPLICATE_VALUE
Duplicate parameter.

MQRCCF_CFIL_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIL_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Inquire Queue Names

The Inquire Queue Names (MQCMD_INQUIRE_Q_NAMES) command inquires a list of queue names that match the generic queue name, and the optional queue type specified.

This PCF is supported on all platforms.

Required parameters:
QName

Optional parameters:
QType

The Inquire Queue Names command, if successful, generates a data response. For details of the Inquire Queue Names response, refer to "Data responses to commands" on page 226.

Required parameters

QName (MQCFST)
Queue name (parameter identifier: MQCA_Q_NAME).

Generic queue names are supported. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string. An asterisk on its own matches all possible names.

The maximum length of the string is MQ_Q_NAME_LENGTH.

Optional parameters

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

If present, this parameter limits the queue names returned to queues of the specified type. If this parameter is not present, queues of all types are eligible. The value may be:

MQQT_ALL
All queue types.

MQQT_LOCAL
Local queue.

MQQT_ALIAS
Alias queue definition.

MQQT_REMOTE
Local definition of a remote queue.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_Q_TYPE_ERROR
Queue type not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Ping Queue Manager

The Ping Queue Manager (MQCMD_PING_Q_MGR) command tests whether the queue manager and its command server is responsive to commands. If the queue manager is responding a positive reply is returned.

This PCF is supported on all platforms.

Required parameters:

None

Optional parameters:

None

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

Reset Channel

The Reset Channel (MQCMD_RESET_CHANNEL) command resets the message sequence number for an MQSeries channel with, optionally, a specified sequence number to be used the next time that the channel is started.

This command can be issued to a channel of type MQCHT_SENDER or MQCHT_RECEIVER. However, if it is issued to a sender (MQCHT_SENDER) channel, the value at both ends (issuing end and receiver end), is reset when the channel is next initiated or resynchronized. The value at both ends is reset to be equal.

If the command is issued to a receiver (MQCHT_RECEIVER) channel, the value at the Sender end is not reset as well; this must be done separately if necessary.

This PCF is supported on all platforms.

Required parameters:

ChannelName

Optional parameters:

MsgSeqNumber

Required parameters

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be reset. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

MsgSeqNumber (MQCFIN)
Message sequence number (parameter identifier: MQIACH_MSG_SEQUENCE_NUMBER).

Specifies the new message sequence number.

The value may be in the range 1-999 999. The default value is one.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175', for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_MSG_SEQ_NUMBER_ERROR
Message sequence number not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Start Channel

The Start Channel (MQCMD_START_CHANNEL) command starts an MQSeries channel.

This command can be issued to a channel of type MQCHT_SENDER or MQCHT_RECEIVER. Under MQSeries for VSE/ESA, starting a channel this way makes it available for use.

This PCF is supported on all platforms.

Required parameters:

ChannelName

Optional parameters:

None

Required parameters

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be started. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_IN_USE
Channel in use.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_CHANNEL_TYPE_ERROR
Channel type not valid.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Start Channel Listener

The Start Channel Listener (MQCMD_START_CHANNEL_LISTENER) command starts an MQSeries TCP listener task.

This command is valid only for TCP transmission protocols. If a Start Channel Listener command is issued for the LU 6.2 protocol, the command is ignored and a successful response is generated.

Required parameters:

None

Optional parameters:

TransportType

Optional parameters

TransportType (MQCFIN)
Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value may be:

MQXPT_LU62
LU 6.2.

MQXPT_TCP
TCP.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_LISTENER_NOT_STARTED
Listener not started.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

Stop Channel

The Stop Channel (MQCMD_STOP_CHANNEL) command stops an MQSeries channel.

This command can be issued to a channel of any type supported by MQSeries for VSE/ESA.

This PCF is supported on all platforms.

Required parameters:

ChannelName

Optional parameters:

Quiesce

Required parameters

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The name of the channel to be stopped. The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

Optional parameters

Quiesce (MQCFIN)
Quiesce channel (parameter identifier: MQIACF_QUIESCE).

Specifies whether the channel should be quiesced or stopped immediately. If this parameter is not present the channel is quiesced. The value may be:

MQQO_YES
Quiesce the channel.

MQQO_NO
Do not quiesce the channel.

Under MQSeries for VSE/ESA, the Quiesce parameter is ignored.

Error codes

In addition to the values for any command shown in section "Error codes applicable to all commands" on page 175, for this command the following may be returned in the response format header:

Reason (MQLONG)

The value may be:

MQRCCF_CFIN_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFIN_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFIN_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_DUPLICATE_PARM
Duplicate parameter.

MQRCCF_CFST_LENGTH_ERROR
Structure length not valid.

MQRCCF_CFST_PARM_ID_ERROR
Parameter identifier is not valid.

MQRCCF_CFST_STRING_LENGTH_ERR
String length not valid.

MQRCCF_CHANNEL_DISABLED
Channel disabled.

MQRCCF_CHANNEL_NOT_ACTIVE
Channel not active.

MQRCCF_CHANNEL_NOT_FOUND
Channel not found.

MQRCCF_PARM_COUNT_TOO_BIG
Parameter count too big.

MQRCCF_PARM_COUNT_TOO_SMALL
Parameter count too small.

MQRCCF_QUIESCE_VALUE_ERROR
Quiesce value not valid.

MQRCCF_STRUCTURE_TYPE_ERROR
Structure type not valid.

Data responses to commands

Escape commands, and commands that request information, if successful, generate data responses. A data response consists of an OK response (as described in "OK response" on page 172) followed by additional structures containing the requested data.

Applications should not depend upon these additional parameter structures being returned in any particular order.

Data responses are generated for these commands:

- Escape
- Inquire Channel
- Inquire Channel Names
- Inquire Queue
- Inquire Queue Manager
- Inquire Queue Names

Escape (Response)

The response to the Escape (MQCMD_ESCAPE) command consists of the response header followed by two parameter structures, one containing the escape type, and the other containing the text response. More than one such message may be issued, depending upon the command contained in the Escape request.

The Command field in the response header MQCFH contains the MQCMD_* command identifier of the text command contained in the EscapeText parameter in the original Escape command. For example, if EscapeText in the original Escape command specified PING QMGR, Command in the response has the value MQCMD_PING_Q_MGR.

If it is possible to determine the outcome of the command, the CompCode in the response header identifies whether the command was successful. The success or otherwise can therefore be determined without the recipient of the response having to parse the text of the response.

If it is not possible to determine the outcome of the command, CompCode in the response header has the value MQCC_UNKNOWN, and Reason is MQRC_NONE.

Always returned:

EscapeType
EscapeText

Returned if requested:

None

Parameters:

EscapeType (MQCFIN)
Escape type (parameter identifier: MQIACF_ESCAPE_TYPE).

The only value supported is:

MQET_MQSC
MQSeries command.

EscapeText (MQCFST)
Escape text (parameter identifier: MQCACF_ESCAPE_TEXT).

A string holding the response to the original command.

Inquire Channel (Response)

The response to the Inquire Channel (MQCMD_INQUIRE_CHANNEL) command consists of the response header followed by the ChannelName structure and the requested combination of attribute parameter structures (where applicable).

This response is supported on all platforms.

Always returned:
ChannelName

Returned if requested:
AllocRetryFastTimer, AllocRetrySlowTimer, AlterationDate, AlterationTime,AllocRetryCount, BatchSize,ChannelDesc, ChannelType, ConnectionName, DataConversion, DiscInterval, DiscRetryCount, MaxMsgLength, MsgExit, MsgUserData, PortNumber, ReceiveExit, ReceiveUserData, SecurityExit, SecurityUserData, SendExit, SendUserData, SeqNumberWrap, SSLCipherSpec, SSLClientAuth, SSLPeerName, TpName, TransportType, XmitQName

Response data:

AlterationDate (MQCFST)
Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date when the information was last altered.

AlterationTime (MQCFST)
Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time when the information was last altered.

AllocRetryCount (MQCFIN)
APPC connection retry count (parameter identifier: MQIACH_ALLOC_RETRY).

AllocRetryFastTimer (MQCFIN)
APPC connection retry fast timer (parameter identifier: MQIACH_ALLOC_FAST_TIMER).

AllocRetrySlowTimer (MQCFIN)
APPC connection retry slow timer (parameter identifier: MQIACH_ALLOC_SLOW_TIMER).

BatchSize (MQCFIN)
Batch size (parameter identifier: MQIACH_BATCH_SIZE).

ChannelDesc (MQCFST)
Channel description (parameter identifier: MQCACH_DESC).

The maximum length of the string is MQ_CHANNEL_DESC_LENGTH.

ChannelName (MQCFST)
Channel name (parameter identifier: MQCACH_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

ChannelType (MQCFIN)
Channel type (parameter identifier: MQIACH_CHANNEL_TYPE).

The value may be:

MQCHT_SENDER
Sender.

MQCHT_RECEIVER
Receiver.

MQCHT_SVRCONN
Server-connection (for use by clients).

ConnectionName (MQCFST)
Connection name (parameter identifier: MQCACH_CONNECTION_NAME).

The maximum length of the string is MQ_CONN_NAME_LENGTH.

DataConversion (MQCFIN)
Whether sender should convert application data (parameter identifier: MQIACH_DATA_CONVERSION).

The value may be:

MQCDC_NO_SENDER_CONVERSION
No conversion by sender.

MQCDC_SENDER_CONVERSION
Conversion by sender.

DiscInterval (MQCFIN)
Disconnection interval (parameter identifier: MQIACH_DISC_INTERVAL).

DiscRetryCount (MQCFIN)
Disconnection retry count (parameter identifier: MQIACH_DISC_RETRY).

MaxMsgLength (MQCFIN)
Maximum message length (parameter identifier: MQIACH_MAX_MSG_LENGTH).

MsgExit (MQCFST)
Message exit name (parameter identifier: MQCACH_MSG_EXIT_NAME).

MsgUserData (MQCFST)
Message exit user data (parameter identifier: MQCACH_MSG_EXIT_USER_DATA).

PortNumber (MQCFIN)
TCP/IP port number (parameter identifier: MQIACH_PORT_NUMBER).

ReceiveExit (MQCFST)
Receive exit name (parameter identifier: MQCACH_RCV_EXIT_NAME).

ReceiveUserData (MQCFST)
Receive exit user data (parameter identifier: MQCACH_RCV_EXIT_USER_DATA).

SecurityExit (MQCFST)
Security exit name (parameter identifier: MQCACH_SEC_EXIT_NAME).

SecurityUserData (MQCFST)
Security exit user data (parameter identifier: MQCACH_SEC_EXIT_USER_DATA).

SendExit (MQCFST)
Send exit name (parameter identifier: MQCACH_SEND_EXIT_NAME).

SendUserData (MQCFST)
Send exit user data (parameter identifier: MQCACH_SEND_EXIT_USER_DATA).

SeqNumberWrap (MQCFIN)
Sequence wrap number (parameter identifier: MQIACH_SEQUENCE_NUMBER_WRAP).

SSLCipherSpec (MQCFIN)
SSL cipher specification (parameter identifier: MQCACH_SSL_CIPHER_SPEC).

SSLClientAuth (MQCFIN)
SSL client authentication (parameter identifier: MQIACH_SSL_CLIENT_AUTH).

The value may be:

MQSCA_OPTIONAL
Client authentication is required.

MQSCA_REQUIRED
Client authentication is optional.

SSLPeerName (MQCFST)
SSL peer name (parameter identifier: MQCACH_SSL_PEER_NAME).

The maximum length of the string is MQ_DISTINGUISHED_NAME_LENGTH.

TpName (MQCFST)
Transaction program name (parameter identifier: MQCACH_TP_NAME).

The maximum length of the string is MQ_TP_NAME_LENGTH.

TransportType (MQCFIN)
Transmission protocol type (parameter identifier: MQIACH_XMIT_PROTOCOL_TYPE).

The value may be:

MQXPT_LU62
LU 6.2.

MQXPT_TCP
TCP.

XmitQName (MQCFST)
Transmission queue name (parameter identifier: MQCACH_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

Inquire Channel Names (Response)

The response to the Inquire Channel Names (MQCMD_INQUIRE_CHANNEL_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified channel name.

This response is supported on all platforms.

Always returned:
ChannelNames

Returned if requested:
None

Response data:

ChannelNames (MQCFSL)
Channel names (parameter identifier: MQCACH_CHANNEL_NAMES).

Inquire Queue (Response)

The response to the Inquire Queue (MQCMD_INQUIRE_Q) command consists of the response header followed by the QName structure and the requested combination of attribute parameter structures.

This PCF is supported on all platforms.

Always returned:

QName

Returned if requested:

AlterationDate, AlterationTime, BaseQName, CICSFileName, CreationDate, CreationTime, InhibitGet, InhibitPut, MaxGlobalLocks, MaxLocalLocks, MaxMsgLength, MaxQDepth, MaxQTriggers, MaxQUsers, QDepthHighEvent, QDepthHighLimit, QDepthLowEvent, QDepthLowLimit, QDepthMaxEvent, QDesc, QServiceInterval, QServiceIntervalEvent, QType, RemoteQMgrName, RemoteQName, Shareability, TriggerChannelName, TriggerControl, TriggerData, TriggerProgramName, TriggerRestart, TriggerTerminalId, TriggerTransactionId, TriggerType, Usage, XmitQName

Response data:

AlterationDate (MQCFST)
Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date when the information was last altered.

AlterationTime (MQCFST)
Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time when the information was last altered.

BaseQName (MQCFST)
Queue name to which the alias resolves (parameter identifier: MQCA_BASE_Q_NAME).

This is the name of a queue that is defined to the local queue manager.

The maximum length of the string is MQ_Q_NAME_LENGTH.

CICSFileName (MQCFST)
CSD file name for queue messages (parameter identifier: MQCA_CICS_FILE_NAME).

The maximum length of the string is MQ_CICS_FILE_NAME_LENGTH.

CreationDate (MQCFST)
Queue creation date (parameter identifier: MQCA_CREATION_DATE).

The maximum length of the string is MQ_CREATION_DATE_LENGTH.

CreationTime (MQCFST)
Creation time (parameter identifier: MQCA_CREATION_TIME).

The maximum length of the string is MQ_CREATION_TIME_LENGTH.

InhibitGet (MQCFIN)
Whether get operations are allowed (parameter identifier: MQIA_INHIBIT_GET).

The value may be:

MQQA_GET_ALLOWED
Get operations are allowed.

MQQA_GET_INHIBITED
Get operations are inhibited.

InhibitPut (MQCFIN)
Whether put operations are allowed (parameter identifier: MQIA_INHIBIT_PUT).

The value may be:

MQQA_PUT_ALLOWED
Put operations are allowed.

MQQA_PUT_INHIBITED
Put operations are inhibited.

MaxGlobalLocks (MQCFIN)
Buffer size for queue manager to manage concurrent queue access (parameter identifier: MQIA_MAX_GLOBAL_LOCKS).

MaxLocalLocks (MQCFIN)
Buffer size for applications to manage concurrent queue access (parameter identifier: MQIA_MAX_LOCAL_LOCKS).

MaxMsgLength (MQCFIN)
Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

MaxQDepth (MQCFIN)
Maximum queue depth (parameter identifier: MQIA_MAX_Q_DEPTH).

MaxQTriggers (MQCFIN)
Maximum number of concurrent trigger instances for a particular queue (parameter identifier: MQIA_MAX_Q_TRIGGERS).

MaxQUsers (MQCFIN)
Maximum number of active opens to any particular queue (parameter identifier: MQIA_Q_USERS).

QDepthHighEvent (MQCFIN)
Controls whether Queue Depth High events are generated (parameter identifier: MQIA_Q_DEPTH_HIGH_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QDepthHighLimit (MQCFIN)
High limit for queue depth (parameter identifier: MQIA_Q_DEPTH_HIGH_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth High event.

QDepthLowEvent (MQCFIN)
Controls whether Queue Depth Low events are generated (parameter identifier: MQIA_Q_DEPTH_LOW_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QDepthLowLimit (MQCFIN)
Low limit for queue depth (parameter identifier: MQIA_Q_DEPTH_LOW_LIMIT).

The threshold against which the queue depth is compared to generate a Queue Depth Low event.

QDepthMaxEvent (MQCFIN)
Controls whether Queue Full events are generated (parameter identifier: MQIA_Q_DEPTH_MAX_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

QDesc (MQCFST)
Queue description (parameter identifier: MQCA_Q_DESC).

The maximum length of the string is MQ_Q_DESC_LENGTH.

QName (MQCFST)
Queue name (parameter identifier: MQCA_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

QServiceInterval (MQCFIN)
Target for queue service interval (parameter identifier: MQIA_Q_SERVICE_INTERVAL).

The service interval used for comparison to generate Queue Service Interval High and Queue Service Interval OK events.

QServiceIntervalEvent (MQCFIN)
Controls whether Service Interval High or Service Interval OK events are generated (parameter identifier: MQIA_Q_SERVICE_INTERVAL_EVENT).

The value may be:

MQQSIE_HIGH
Queue Service Interval High events enabled.

MQQSIE_OK
Queue Service Interval OK events enabled.

MQQSIE_NONE
No queue service interval events enabled.

QType (MQCFIN)
Queue type (parameter identifier: MQIA_Q_TYPE).

The value may be:

MQQT_ALIAS
Alias queue definition.

MQQT_LOCAL
Local queue.

MQQT_REMOTE
Local definition of a remote queue.

RemoteQName (MQCFST)
Name of remote queue as known locally on the remote queue manager (parameter identifier: MQCA_REMOTE_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

RemoteQMgrName (MQCFST)
Name of remote queue manager (parameter identifier: MQCA_REMOTE_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

Shareability (MQCFIN)
Whether queue can be shared (parameter identifier: MQIA_SHAREABILITY).

The value may be:

MQQA_SHAREABLE
Queue is shareable.

MQQA_NOT_SHAREABLE
Queue is not shareable.

TriggerChannelName (MQCFST)
Channel name for MCA trigger process (parameter identifier: MQCA_TRIGGER_CHANNEL_NAME).

The maximum length of the string is MQ_CHANNEL_NAME_LENGTH.

TriggerControl (MQCFIN)
Trigger control (parameter identifier: MQIA_TRIGGER_CONTROL).

The value may be:

MQTC_OFF
Trigger messages not required.

MQTC_ON
Trigger messages required.

TriggerData (MQCFST)
Trigger data (parameter identifier: MQCA_TRIGGER_DATA).

The maximum length of the string is MQ_PROCESS_USER_DATA_LENGTH.

TriggerProgramName (MQCFST)
Program name for trigger process (parameter identifier: MQCA_TRIGGER_PROGRAM_NAME).

The maximum length of the string is MQ_TRIGGER_PROGRAM_NAME_LENGTH.

TriggerRestart (MQCFIN)
Indicator for the reactivation of a trigger process (parameter identifier: MQIA_TRIGGER_RESTART).

The value may be:

MQTRIGGER_RESTART_NO
Do not reactivate trigger process.

MQTRIGGER_RESTART_YES
Reactivate trigger process.

TriggerTerminalId (MQCFST)
Terminal identifier for trigger process (parameter identifier: MQCA_TRIGGER_TERM_ID).

The maximum length of the string is MQ_TRIGGER_TERM_ID_LENGTH.

TriggerTransactionId (MQCFST)
Transaction identifier for trigger process (parameter identifier: MQCA_TRIGGER_TRANS_ID).

The maximum length of the string is MQ_TRIGGER_TRANS_ID_LENGTH.

TriggerType (MQCFIN)
Trigger type (parameter identifier: MQIA_TRIGGER_TYPE).

The value may be:

MQTT_NONE
No trigger messages.

MQTT_FIRST
Trigger message when queue depth goes from 0 to 1.

MQTT_EVERY
Trigger message for every message.

Usage (MQCFIN)
Usage (parameter identifier: MQIA_USAGE).

The value may be:

MQUS_NORMAL
Normal usage.

MQUS_TRANSMISSION
Transmission queue.

XmitQName (MQCFST)
Transmission queue name (parameter identifier: MQCA_XMIT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

Inquire Queue Manager (Response)

The response to the Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) command consists of the response header followed by the QMgrName structure and the requested combination of attribute parameter structures.

This response is supported on all platforms.

Always returned:
QMgrName

Returned if requested:
AlterationDate, AlterationTime, AuthorityEvent, BatchInterfaceAutoStart, BatchInterfaceId, CodedCharSetId, CommandInputQName, CommandLevel, CommandReplyQName, CommandServerAutoStart, CommandServerDataConversion, CommandServerDeadLetterQ, DeadLetterQName, DistLists, InhibitEvent, ListenerPortNumber, LocalEvent, MaxGlobalLocks, MaxHandles, MaxLocalLocks, MaxMsgLength, MaxOpenQ, MaxQDepth, MaxQUsers, MonitorInterval, MonitorQName, PerformanceEvent, Platform, QMgrDesc, RemoteEvent, SSLKeyLibraryMember, SSLKeyLibraryName, StartStopEvent, SyncPoint, SystemLogQName

Response data:

AlterationDate (MQCFST)
Alteration date (parameter identifier: MQCA_ALTERATION_DATE).

The date when the information was last altered.

AlterationTime (MQCFST)
Alteration time (parameter identifier: MQCA_ALTERATION_TIME).

The time when the information was last altered.

AuthorityEvent (MQCFIN)
Controls whether authorization (Not Authorized) events are generated (parameter identifier: MQIA_AUTHORITY_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

BatchInterfaceAutoStart (MQCFIN)
Indicator for the automatic activation of the batch interface (parameter identifier: MQIA_BATCH_INTERFACE_AUTO).

The value may be:

MQAUTO_START_NO
Do not automatically start the batch interface.

MQAUTO_START_YES
Automatically start the batch interface.

BatchInterfaceId (MQCFST)
Batch interface identifier (parameter identifier: MQCA_BATCH_INTERFACE_ID).

The maximum length of the string is MQ_BATCH_INTERFACE_ID_LENGTH.

CodedCharSetId (MQCFIN)
Coded character set identifier (parameter identifier: MQIA_CODED_CHAR_SET_ID).

CommandInputQName (MQCFST)
Command input queue name (parameter identifier: MQCA_COMMAND_INPUT_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

CommandLevel (MQCFIN)
Command level supported by queue manager (parameter identifier: MQIA_COMMAND_LEVEL).

For MQSeries for VSE/ESA, the value is MQCMDL_LEVEL_211.

CommandReplyQName (MQCFST)
MQSC reply queue name (parameter identifier: MQCA_COMMAND_REPLY_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

CommandServerAutoStart (MQCFIN)
Indicator for the automatic activation of the PCF command server (parameter identifier: MQIA_CMD_SERVER_AUTO).

The value may be:

MQAUTO_START_NO
Do not automatically start the PCF command server.

MQAUTO_START_YES
Automatically start the PCF command server.

CommandServerDataConversion (MQCFIN)
Indicator for the data conversion of PCF messages (parameter identifier MQIA_CMD_SERVER_CONVERT_MSG).

The value may be:

MQCSRV_CONVERT_NO
Do not convert PCF messages.

MQCSRV_CONVERT_YES
Convert PCF messages.

CommandServerDeadLetterQ (MQCFIN)
Indicator for the storage of undeliverable PCF reply messages (parameter identifier: MQIA_CMD_SERVER_DLQ_MSG).

The value bay be:

MQCSRV_DLQ_NO
Do not store undeliverable PCF replies to DLQ.

MQCSRV_DLQ_YES
Store undeliverable PCF replies to DLQ.

DeadLetterQName (MQCFST)
Dead letter (undelivered message) queue name (parameter identifier: MQCA_DEAD_LETTER_Q_NAME).

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The maximum length of the string is MQ_Q_NAME_LENGTH.

DistLists (MQCFIN)
Distribution list support (parameter identifier: MQIA_DIST_LISTS).

For MQSeries for VSE/ESA, the value is MQDL_NOT_SUPPORTED.

InhibitEvent (MQCFIN)
Controls whether inhibit (Inhibit Get and Inhibit Put) events are generated (parameter identifier: MQIA_INHIBIT_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

ListenerPortNumber (MQCFIN)
Port number for TCP/IP Listener process (parameter identifier: MQIA_LISTENER_PORT_NUMBER).

LocalEvent (MQCFIN)
Controls whether local error events are generated (parameter identifier: MQIA_LOCAL_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

MaxGlobalLocks (MQCFIN)
Buffer size for queue manager to manage concurrent queue access (parameter identifier: MQIA_MAX_GLOBAL_LOCKS).

MaxHandles (MQCFIN)
Maximum number of handles (parameter identifier: MQIA_MAX_HANDLES).

Specifies the maximum number of MQI connections that will be handled by the queue manager at any one time.

The value may be in the range 1 through 1000.

MaxLocalLocks (MQCFIN)
Buffer size for applications to manage concurrent queue access (parameter identifier: MQIA_MAX_LOCAL_LOCKS).

MaxMsgLength (MQCFIN)
Maximum message length (parameter identifier: MQIA_MAX_MSG_LENGTH).

MaxOpenQ (MQCFIN)
Maximum number of concurrently open queues (parameter identifier: MQIA_MAX_OPEN_Q).

MaxQDepth (MQCFIN)
Maximum queue depth (parameter identifier: MQIA_MAX_Q_DEPTH).

MaxQUsers (MQCFIN)
Maximum number of active opens to any particular queue (parameter identifier: MQIA_Q_USERS).

MonitorInterval (MQCFIN)
Queue manager housekeeping process interval (parameter identifier: MQIA_MONITOR_INTERVAL).

MonitorQName (MQCFST)
MQI monitor queue name (parameter identifier: MQCA_MONITOR_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

PerformanceEvent (MQCFIN)
Controls whether performance-related events are generated (parameter identifier: MQIA_PERFORMANCE_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

Platform (MQCFIN)
Platform on which the queue manager resides (parameter identifier: MQIA_PLATFORM).

For MQSeries for VSE/ESA, the value is PL_VSE.

QmgrDesc (MQCFST)
Queue manager description (parameter identifier: MQCA_Q_MGR_DESC).

The maximum length of the string is MQ_Q_MGR_DESC_LENGTH.

QMgrName (MQCFST)
Name of local queue manager (parameter identifier: MQCA_Q_MGR_NAME).

The maximum length of the string is MQ_Q_MGR_NAME_LENGTH.

RemoteEvent (MQCFIN)
Controls whether remote error events are generated (parameter identifier: MQIA_REMOTE_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

SSLKeyLibraryMember (MQCFST)
SSL key library member name (parameter identifier: MQCA_SSL_KEY_MEMBER).

The maximum length of the string is MQ_SSL_KEY_MEMBER_LENGTH.

SSLKeyLibraryName (MQCFST)
SSL key library name (parameter identifier: MQCA_SSL_KEY_LIBRARY).

The maximum length of the string is MQ_SSL_KEY_LIBRARY_LENGTH.

StartStopEvent (MQCFIN)
Controls whether start and stop events are generated (parameter identifier: MQIA_START_STOP_EVENT).

The value may be:

MQEVR_DISABLED
Event reporting disabled.

MQEVR_ENABLED
Event reporting enabled.

SyncPoint (MQCFIN)
Syncpoint availability (parameter identifier: MQIA_SYNCPOINT).

For MQSeries for VSE/ESA, the value is MQSP_NOT_AVAILABLE.

SystemLogQName
System log queue name (parameter identifier: MQCA_SYSTEM_LOG_Q_NAME).

The maximum length of the string is MQ_Q_NAME_LENGTH.

Inquire Queue Names (Response)

The response to the Inquire Queue Names (MQCMD_INQUIRE_Q_NAMES) command consists of the response header followed by a single parameter structure giving zero or more names that match the specified queue name.

This response is supported on all platforms.

Always returned:
QNames

Returned if requested:
None

Response data:

QNames (MQCFSL)
Queue names (parameter identifier: MQCACF_Q_NAMES).

Structures used for commands and responses

Commands and responses consist of a PCF header (MQCFH) structure followed by zero or more parameter structures. Each of these is one of:

PCF integer parameter (MQCFIN)
PCF string parameter (MQCFST)
PCF integer list parameter (MQCFIL)
PCF string list parameter (MQCFSL)

This section defines these parameter structures.

MQCFH - PCF header

The MQCFH structure describes the information that is present at the start of the message data of a command message, or a response to a command message. In either case, the message descriptor Format field is MQFMT_ADMIN.

Type (MQLONG)
Structure type. This indicates the content of the message. These are valid:

MQCFT_COMMAND
Message is a command.

MQCFT_RESPONSE
Message is a response to a command.

StrucLength (MQLONG)
Structure length. This is the length in bytes of the MQCFH structure. The value must be:

MQCFH_STRUC_LENGTH
Length of command format header structure.

The initial value of this field is MQCFH_STRUC_LENGTH.

Version (MQLONG)
Structure version number. The value must be:

MQCFH_VERSION_1
Version number for command format header structure.

Command (MQLONG)
Command identifier. For a command message, this identifies the function to be performed. For a response message, it identifies the command to which this is the reply. These are valid:

MQCMD_CHANGE_Q_MGR
Change queue manager.

MQCMD_INQUIRE_Q_MGR
Inquire queue manager.

MQCMD_PING_Q_MGR
Ping queue manager.

MQCMD_CHANGE_Q
Change queue.

MQCMD_COPY_Q
Copy queue.

MQCMD_CREATE_Q
Create queue.

MQCMD_DELETE_Q
Delete queue.

MQCMD_INQUIRE_Q
Inquire queue.

MQCMD_INQUIRE_Q_NAMES
Inquire queue names.

MQCMD_CHANGE_CHANNEL
Change channel.

MQCMD_COPY_CHANNEL
Copy channel.

MQCMD_CREATE_CHANNEL
Create channel.

MQCMD_DELETE_CHANNEL
Delete channel.

MQCMD_INQUIRE_CHANNEL
Inquire channel.

MQCMD_RESET_CHANNEL
Reset channel.

MQCMD_START_CHANNEL
Start channel.

MQCMD_STOP_CHANNEL
Stop channel.

MQCMD_START_CHANNEL_LISTENER
Start channel listener.

MQCMD_INQUIRE_CHANNEL_NAMES
Inquire channel names.

MQCMD_ESCAPE
Escape.

MsgSeqNumber (MQLONG)
Message sequence number. This is the sequence number of the message within a group of related messages. For a command, this field must have the value one (because a command is always contained within a single

message). For a response, the field has the value one for the first (or only) response to a command, and increases by one for each successive response to that command.

The last (or only) message in a group has the MQCFC_LAST flag set in the Control field.

The initial value of this field is 1.

Control (MQLONG)
Control options. These are valid:

MQCFC_LAST
Last message in the group.

For a command, this value must always be set.

MQCFC_NOT_LAST
Not the last message in the group.

The initial value of this field is MQCFC_LAST.

CompCode (MQLONG)
Completion code. This field is meaningful only for a response; its value is not significant for a command. These are possible:

MQCC_OK
Command completed successfully.

MQCC_WARNING
Command completed with warning.

MQCC_FAILED
Command failed.

MQCC_UNKNOWN
Whether command succeeded is not known.

The initial value of this field is MQCC_OK.

Reason (MQLONG)
Reason code qualifying completion code. This field is meaningful only for a response; its value is not significant for a command.

The possible reason codes that could be returned in response to a command are listed at the end of each command format described in "Definitions of the PCFs" on page 174.

The initial value of this field is MQRC_NONE.

ParameterCount (MQLONG)
Count of parameter structures. This is the number of parameter structures (MQCFIL, MQCFIN, MQCFSL, and MQCFST) that follow the MQCFH structure. The value of this field is zero or greater.

The initial value of this field is 0.

C language declaration

The C language declaration for the MQCFH data structure is:

```
typedef struct tagMQCFH {
        MQLONG  Type;           /* Structure type */
        MQLONG  StrucLength;    /* Structure length */
        MQLONG  Version;        /* Structure version number */
        MQLONG  Command;        /* Command identifier */
        MQLONG  MsgSeqNumber;   /* Message sequence number */
        MQLONG  Control;        /* Control options */
```

```
        MQLONG  CompCode;        /* Completion code */
        MQLONG  Reason;          /* Reason code qualifying completion code */
        MQLONG  ParameterCount;  /* Count of parameter structures */
} MQCFH;
```

COBOL language declaration

The COBOL language declaration for the MQCFH data structure is:

```
**   MQCFH structure
10 MQCFH.
**    Structure type
   15 MQCFH-TYPE           PIC S9(9) BINARY.
**    Structure length
   15 MQCFH-STRUCLENGTH    PIC S9(9) BINARY.
**    Structure version number
   15 MQCFH-VERSION        PIC S9(9) BINARY.
**    Command identifier
   15 MQCFH-COMMAND        PIC S9(9) BINARY.
**    Message sequence number
   15 MQCFH-MSGSEQNUMBER   PIC S9(9) BINARY.
**    Control options
   15 MQCFH-CONTROL        PIC S9(9) BINARY.
**    Completion code
   15 MQCFH-COMPCODE       PIC S9(9) BINARY.
**    Reason code qualifying completion code
   15 MQCFH-REASON         PIC S9(9) BINARY.
**    Count of parameter structures
   15 MQCFH-PARAMETERCOUNT PIC S9(9) BINARY.
```

PL/I language declaration

The PL/I language declaration for the MQCFH data structure is:

```
dcl
 1 MQCFH based,
  3 Type           fixed bin(31), /* Structure type */
  3 StrucLength    fixed bin(31), /* Structure length */
  3 Version        fixed bin(31), /* Structure version number */
  3 Command        fixed bin(31), /* Command identifier */
  3 MsgSeqNumber   fixed bin(31), /* Message sequence number */
  3 Control        fixed bin(31), /* Control options */
  3 CompCode       fixed bin(31), /* Completion code */
  3 Reason         fixed bin(31), /* Reason code qualifying completion code */
  3 ParameterCount fixed bin(31); /* Count of parameter structures */
```

MQCFIN - PCF integer parameter

The MQCFIN structure describes an integer parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

Type (MQLONG)

Structure type. This indicates that the structure is a MQCFIN structure describing an integer parameter. The value must be:

MQCFT_INTEGER

Structure defining an integer.

The initial value of this field is MQCFT_INTEGER.

StrucLength (MQLONG)

Structure length. This is the length in bytes of the MQCFIN structure. The value must be:

MQCFIN_STRUC_LENGTH

Length of command format integer-parameter structure.

The initial value of this field is MQCFIN_STRUC_LENGTH.

Parameter (MQLONG)
Parameter identifier. This identifies the parameter whose value is contained in the structure. The values that can occur in this field depend on the value of the Command field in the MQCFH structure; see "MQCFH - PCF header" on page 240 for details.

The initial value of this field is 0.

Value (MQLONG)
Parameter value. This is the value of the parameter identified by the Parameter field.

The initial value of this field is 0.

C language declaration

The C language declaration for the MQCFIN data structure is:

```
typedef struct tagMQCFIN {
        MQLONG  Type;         /* Structure type */
        MQLONG  StrucLength;  /* Structure length */
        MQLONG  Parameter;    /* Parameter identifier */
        MQLONG  Value;        /* Parameter value */
} MQCFIN;
```

COBOL language declaration

The COBOL language declaration for the MQCFIN data structure is:

```
**   MQCFIN structure
10 MQCFIN.
**    Structure type
   15 MQCFIN-TYPE        PIC S9(9) BINARY.
**    Structure length
   15 MQCFIN-STRUCLENGTH PIC S9(9) BINARY.
**    Parameter identifier
   15 MQCFIN-PARAMETER   PIC S9(9) BINARY.
**    Parameter value
   15 MQCFIN-VALUE       PIC S9(9) BINARY.
```

PL/I language declaration

The PL/I language declaration for the MQCFIN data structure is:

```
dcl
 1 MQCFIN based,
  3 Type        fixed bin(31), /* Structure type */
  3 StrucLength fixed bin(31), /* Structure length */
  3 Parameter   fixed bin(31), /* Parameter identifier */
  3 Value       fixed bin(31); /* Parameter value */
```

MQCFST - PCF string parameter

The MQCFST structure describes a string parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The structure ends with a variable-length character string; see the String field below for further details.

Type (MQLONG)
Structure type. This indicates that the structure is an MQCFST structure describing a string parameter. The value must be:

MQCFT_STRING
Structure defining a string.

The initial value of this field is MQCFT_STRING.

StrucLength (MQLONG)
Structure length. This is the length in bytes of the MQCFST structure, including the String field). The length must be a multiple of four, and must be sufficient to contain the string; any bytes between the end of the string and the length defined by the StrucLength field are not significant.

The following constant gives the length of the fixed part of the structure, that is the length excluding the String field:

MQCFST_STRUC_LENGTH_FIXED
Length of fixed part of command format string-parameter structure.

The initial value of this field is MQCFST_STRUC_LENGTH_FIXED.

Parameter (MQLONG)
Parameter identifier. This identifies the parameter whose value is contained in the structure. The values that can occur in this field depend on the value of the Command field in the MQCFH structure; see "MQCFH - PCF header" on page 240 for details.

The initial value of this field is 0.

CodedCharSetId (MQLONG)
Coded character set identifier. This specifies the coded character set identifier of the data in the String field. This special value can be used:

MQCCSI_DEFAULT
Default coded character set identifier.

Character data is in the character set defined by the CodedCharSetId field in the message descriptor MQMD.

The initial value of this field is MQCCSI_DEFAULT.

StringLength (MQLONG)
Length of string. This is the length in bytes of the data in the String field; it must be zero or greater. This length need not be a multiple of four.

The initial value of this field is 0.

String (MQCHAR××StringLength)
String value. This is the value of the parameter identified by the Parameter field:

- In MQFMT_ADMIN command messages, if the specified string is shorter than the standard length of the parameter, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, those characters in excess of the standard length must be blanks.
- In MQFMT_ADMIN response messages, string parameters are returned padded with blanks to the standard length of the parameter.

The string can contain any characters that are in the character set defined by CodedCharSetId, and that are valid for the parameter identified by Parameter.

Note: In the MQCFST structure, a null character in the string is treated as normal data, and does not act as a delimiter for the string. This means that when a receiving application reads a MQFMT_ADMIN message, the receiving application receives all of the data specified

by the sending application. The data may, of course, have been converted between character sets (for example, by the receiving application specifying the MQGMO_CONVERT option on the MQGET call).

In contrast, when the queue manager reads an MQFMT_ADMIN message from the command input queue, the queue manager processes the data as though it had been specified on an MQI call. This means that within the string, the first null and the characters following it (up to the end of the string) are treated as blanks.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure should be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL and PL/I programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, the user should include MQCFST in a larger structure, and declare additional field(s) following MQCFST, to represent the String field as required.

In C, the initial value of this field is the null string.

C language declaration

The C language declaration for the MQCFST data structure is:

```
typedef struct tagMQCFST {
        MQLONG  Type;            /* Structure type */
        MQLONG  StrucLength;     /* Structure length */
        MQLONG  Parameter;       /* Parameter identifier */
        MQLONG  CodedCharSetId;  /* Coded character set identifier */
        MQLONG  StringLength;    /* Length of string */
        MQCHAR  String[1];       /* String value - first character */
} MQCFST;
```

COBOL language declaration

The COBOL language declaration for the MQCFST data structure is:

```
**   MQCFST structure
10 MQCFST.
**    Structure type
   15 MQCFST-TYPE           PIC S9(9) BINARY.
**    Structure length
   15 MQCFST-STRUCLENGTH    PIC S9(9) BINARY.
**    Parameter identifier
   15 MQCFST-PARAMETER      PIC S9(9) BINARY.
**    Coded character set identifier
   15 MQCFST-CODEDCHARSETID PIC S9(9) BINARY.
**    Length of string
   15 MQCFST-STRINGLENGTH   PIC S9(9) BINARY.
```

PL/I language declaration

The PL/I language declaration for the MQCFST data structure is:

```
dcl
 1 MQCFST based,
  3 Type           fixed bin(31), /* Structure type */
  3 StrucLength    fixed bin(31), /* Structure length */
  3 Parameter      fixed bin(31), /* Parameter identifier */
  3 CodedCharSetId fixed bin(31), /* Coded character set identifier */
  3 StringLength   fixed bin(31); /* Length of string */
```

MQCFIL - PCF integer list parameter

The MQCFIL structure describes an integer-list parameter in a message that is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The structure ends with a variable-length array of integers; see the Values field below for further details.

Type (MQLONG)
Structure type. This indicates that the structure is an MQCFIL structure describing an integer-list parameter. The value must be:

MQCFT_INTEGER_LIST
Structure defining an integer list.

The initial value of this field is MQCFT_INTEGER_LIST.

StrucLength (MQLONG)
Structure length. This is the length in bytes of the MQCFIL structure, including the Values field). The length must be a multiple of four, and must be sufficient to contain the array; any bytes between the end of the array and the length defined by the StrucLength field are not significant.

The following constant gives the length of the fixed part of the structure, that is the length excluding the Values field:

MQCFIL_STRUC_LENGTH_FIXED
Length of fixed part of command format integer-list parameter structure.

The initial value of this field is MQCFIL_STRUC_LENGTH_FIXED.

Parameter (MQLONG)
Parameter identifier. This identifies the parameter whose values are contained in the structure. The values that can occur in this field depend on the value of the Command field in the MQCFH structure; see "MQCFH - PCF header" on page 240 for details.

The initial value of this field is 0.

Count (MQLONG)
Count of parameter values. This is the number of elements in the Values array; it must be zero or greater.

The initial value of this field is 0.

Values (MQLONG×Count)
Parameter values. This is an array of values for the parameter identified by the Parameter field. For example, for MQIACF_Q_ATTRS, this is a list of attribute selectors (MQCA_* and MQIA_* values).

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure should be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL and PL/I programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, the user should include MQCFIN in a larger structure, and declare additional field(s) following MQCFIN, to represent the Values field as required.

In C, the initial value of this field is a single 0.

C language declaration

The C language declaration for the MQCFIL data structure is:

```
typedef struct tagMQCFIL {
        MQLONG  Type;         /* Structure type */
        MQLONG  StrucLength;  /* Structure length */
        MQLONG  Parameter;    /* Parameter identifier */
        MQLONG  Count;        /* Count of parameter values */
        MQLONG  Values[1];    /* Parameter values - first element */
} MQCFIL;
```

COBOL language declaration

The COBOL language declaration for the MQCFIL data structure is:

```
**   MQCFIL structure
10 MQCFIL.
**    Structure type
   15 MQCFIL-TYPE        PIC S9(9) BINARY.
**    Structure length
   15 MQCFIL-STRUCLENGTH PIC S9(9) BINARY.
**    Parameter identifier
   15 MQCFIL-PARAMETER   PIC S9(9) BINARY.
**    Count of parameter values
   15 MQCFIL-COUNT       PIC S9(9) BINARY.
```

PL/I language declaration

The PL/I language declaration for the MQCFIL data structure is:

```
dcl
 1 MQCFIL based,
  3 Type        fixed bin(31), /* Structure type */
  3 StrucLength fixed bin(31), /* Structure length */
  3 Parameter   fixed bin(31), /* Parameter identifier */
  3 Count       fixed bin(31); /* Count of parameter values */
```

MQCFSL - PCF string list parameter

The MQCFSL structure describes a string-list parameter in a message which is a command or a response to a command. In either case, the format name in the message descriptor is MQFMT_ADMIN.

The structure ends with a variable-length array of character strings; see the Strings field below for further details.

Type (MQLONG)

Structure type. This indicates that the structure is an MQCFSL structure describing a string-list parameter. The value must be:

MQCFT_STRING_LIST

Structure defining a string list.

The initial value of this field is MQCFT_STRING_LIST.

StrucLength (MQLONG)

Structure length. This is the length in bytes of the MQCFSL structure, including the Strings field). The length must be a multiple of four, and must be sufficient to contain all of the strings; any bytes between the end of the strings and the length defined by the StrucLength field are not significant.

The following constant gives the length of the fixed part of the structure, that is the length excluding the Strings field:

MQCFSL_STRUC_LENGTH_FIXED
Length of fixed part of command format string-list parameter structure.

The initial value of this field is MQCFSL_STRUC_LENGTH_FIXED.

Parameter (MQLONG)
Parameter identifier. This identifies the parameter whose values are contained in the structure. The values that can occur in this field depend on the value of the Command field in the MQCFH structure; see "MQCFH - PCF header" on page 240 for details.

The initial value of this field is 0.

CodedCharSetId (MQLONG)
Coded character set identifier. This specifies the coded character set identifier of the data in the Strings field. This special value can be used:

MQCCSI_DEFAULT
Default coded character set identifier. Character data is in the character set defined by the CodedCharSetId field in the message descriptor MQMD.

The initial value of this field is MQCCSI_DEFAULT.

Count (MQLONG)
Count of parameter values. This is the number of strings present in the Strings field; it must be zero or greater.

The initial value of this field is 0.

StringLength (MQLONG)
Length of one string. This is the length in bytes of one parameter value, that is the length of one string in the Strings field; all of the strings are this length. The length must be zero or greater, and need not be a multiple of four.

The initial value of this field is 0.

Strings (MQCHAR×StringLength×Count)
String values. This is a set of string values for the parameter identified by the Parameter field. The number of strings is given by the Count field, and the length of each string is given by the StringLength field. The strings are concatenated together, with no bytes skipped between adjacent strings. The total length of the strings is the length of one string multiplied by the number of strings present (that is, StringLength×Count).

In MQFMT_ADMIN command messages, if the specified string is shorter than the standard length of the parameter, the omitted characters are assumed to be blanks. If the specified string is longer than the standard length, those characters in excess of the standard length must be blanks.

In MQFMT_ADMIN response messages, string parameters are returned padded with blanks to the standard length of the parameter.

In all cases, StringLength gives the length of the string actually present in the message.

The strings can contain any characters that are in the character set defined by CodedCharSetId, and that are valid for the parameter identified by Parameter.

Note: In the MQCFSL structure, a null character in a string is treated as normal data, and does not act as a delimiter for the string. This means that when a receiving application reads a MQFMT_ADMIN message, the receiving application receives all of the data specified by the sending application. The data may, of course, have been converted between character sets (for example, by the receiving application specifying the MQGMO_CONVERT option on the MQGET call).

In contrast, when the queue manager reads an MQFMT_ADMIN message from the command input queue, the queue manager processes the data as though it had been specified on an MQI call. This means that within each string, the first null and the characters following it (up to the end of the string) are treated as blanks.

The way that this field is declared depends on the programming language:

- For the C programming language, the field is declared as an array with one element. Storage for the structure should be allocated dynamically, and pointers used to address the fields within it.
- For the COBOL and PL/I programming languages, the field is omitted from the structure declaration. When an instance of the structure is declared, the user should include MQCFSL in a larger structure, and declare additional field(s) following MQCFSL, to represent the Strings field as required.

In C, the initial value of this field is the null string.

C language declaration

The C language declaration for the MQCFSL data structure is:

```
typedef struct tagMQCFSL {
        MQLONG  Type;            /* Structure type */
        MQLONG  StrucLength;     /* Structure length */
        MQLONG  Parameter;       /* Parameter identifier */
        MQLONG  CodedCharSetId;  /* Coded character set identifier */
        MQLONG Count;           /* Count of parameter values */
        MQLONG  StringLength;    /* Length of one string */
        MQCHAR  Strings[1];      /* String values - first character */
} MQCFSL;
```

COBOL language declaration

The COBOL language declaration for the MQCFSL data structure is:

```
**   MQCFSL structure
10 MQCFSL.
**    Structure type
   15 MQCFSL-TYPE           PIC S9(9) BINARY.
**    Structure length
   15 MQCFSL-STRUCLENGTH    PIC S9(9) BINARY.
**    Parameter identifier
   15 MQCFSL-PARAMETER      PIC S9(9) BINARY.
**    Coded character set identifier
   15 MQCFSL-CODEDCHARSETID PIC S9(9) BINARY.
**    Count of parameter values
   15 MQCFSL-COUNT          PIC S9(9) BINARY.
**    Length of one string
   15 MQCFSL-STRINGLENGTH   PIC S9(9) BINARY.
```

PL/I language declaration

The PL/I language declaration for the MQCFSL data structure is:

```
dcl
 1 MQCFSL based,
  3 Type           fixed bin(31), /* Structure type */
  3 StrucLength    fixed bin(31), /* Structure length */
  3 Parameter      fixed bin(31), /* Parameter identifier */
  3 CodedCharSetId fixed bin(31), /* Coded character set identifier */
  3 Count          fixed bin(31), /* Count of parameter values */
  3 StringLength   fixed bin(31); /* Length of one string */
```

MQCFSL - PCF string list parameter

Chapter 9. MQSeries commands

MQSeries commands (MQSC) provide a uniform method of issuing human-readable commands on MQSeries platforms.

This chapter describes:
- Rules for using MQSeries commands
- Issuing MQSeries commands
- The MQSeries commands

| Review section "Features" on page 8 for prerequisites for this feature.

Rules for using MQSeries commands

You should observe the following rules when using MQSeries commands:

- Each command starts with a primary parameter (a verb), and this is followed by a secondary parameter (a noun). This is then followed by the name of the object (in parentheses) if there is one, which there is on most commands. Following that, parameters can usually occur in any order; if a parameter has a corresponding value, the value must occur directly after the parameter to which it relates.
- Keywords, parentheses, and values can be separated by any number of blanks. There must be at least one blank immediately preceding each parameter.
- Any number of blanks can occur at the beginning or end of the command, and between parameters, punctuation, and values. For example, this command is valid:

  ```
  ALTER QLOCAL ('Account')TRIGDPTH (1)
  ```

  Blanks within a pair of quotation marks are significant.
- Repeated parameters are not allowed.
- Strings that contain non-alphanumeric characters must be enclosed in single quotation marks.
- A string containing no characters (that is, two single quotation marks with no space in between) is not valid.
- A left parenthesis followed by a right parenthesis, with no significant information in between. For example `NAME ()` is not valid.
- Keywords are not case sensitive — AltER, alter, and ALTER are all acceptable. Names that are not contained within quotation marks are converted to uppercase.
- Synonyms are defined for some parameters. For example, DEF is always a synonym for DEFINE, so DEF QLOCAL is valid. Synonyms are not, however, just minimum strings; DEFI is not a valid synonym for DEFINE.

  Note: There is no synonym for the DELETE parameter. This is to avoid accidental deletion of objects when using DEF, the synonym for DEFINE.

The following characters have special meaning when you build MQSeries commands:

© Copyright IBM Corp. 1995, 2004

Table 9. MQSC special characters

Character	Description
blank	Blanks are used as separators. Multiple blanks are equivalent to a single blank, except in strings that have quotation marks (') round them.
'	A single quotation mark indicates the beginning or end of a string. MQSeries leaves all characters that have quotation marks round them exactly as they are entered. The containing quotation marks are not included when calculating the length of the string.
''	Two quotation marks together inside a string are treated by MQSeries as one quotation mark, and the string is not terminated. The double quotation marks are treated as one character when calculating the length of the string.
(	An open parenthesis indicates the beginning of a parameter list.
)	A close parenthesis indicates the end of a parameter list.

MQSeries for VSE/ESA does not support wildcards in MQSeries commands.

Issuing MQSeries commands

With MQSeries for VSE/ESA, MQSeries commands are issued from a batch job, or via PCF Escape commands. MQSeries for VSE/ESA provides a batch utility program (MQPMQSC) which can be used to issue MQSeries commands from batch.

MQSC utility program

The MQSC utility program (MQPMQSC) uses the MQSeries for VSE/ESA batch interface. Consequently, to issue MQSeries commands using the MQPMQSC program, the batch interface, and the MQSeries queue manager, must be active in CICS.

The MQPMQSC program reads MQSeries commands from SYSIPT, converts them into PCF Escape messages and puts them on the system command queue. Responses to MQSeries commands issued this way are sent to the system reply queue. Both the system command queue and the system reply queue are specified as part of the queue manager's global system definition.

The system command and reply queues can be displayed and modified using PCF commands, or the MQMT transaction, option 1.1, followed by PF9:

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
14:47:42                   Global System Definition                 CIC1
MQMMSYS                     Communications Settings                 A000

    TCP/IP settings                          Batch Interface settings
    TCP/IP listener port : 01414             Batch Int. identifier: MQBISERV
    Licensed clients . . : 00000             Batch Int. auto-start: Y
    Adopt MCA  . . . . . : Y
    Adopt MCA Check  . . : Y

    SSL parameters
    Key-ring sublibrary  : PRD2.SSLKEYS
    Key-ring member  . . : MQVSEKEY

    PCF parameters
    System command queue : SYSTEM.ADMIN.COMMAND.QUEUE
    System reply queue . : SYSTEM.ADMIN.REPLY.QUEUE
    Cmd Server auto-start: Y
    Cmd Server convert . : N
    Cmd Server DLQ store : Y

Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 59. System command and reply queues

Since MQSeries commands are issued as PCF Escape messages, the PCF command server (transaction MQCS) must be active in CICS. The command server processes the PCF Escape messages as they arrive on the system command queue, and places PCF Escape responses on the system reply queue. If the command server is not active, the MQPMQSeries command will timeout, and issue an appropriate error response.

The MQPMQSC program reports the results of the MQSeries commands to SYSLST.

MQPMQSC sample JCL

The MQPMQSC program is provided with MQSeries for VSE/ESA and resides in the installation library (default PRD2.MQSERIES). The program should be run in a batch partition.

The following sample JCL illustrates how an MQSeries command (ALTER QLOCAL) can be issued from a batch job:

```
// JOB MQSCJOB
// SETPARM MQBISRV='MQBISRV2'
// LIBDEF *,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
// EXEC MQPMQSC,SIZE=AUTO
ALTER QLOCAL('ANYQ') GET(DISABLED)
/*
/&
```

In this sample, the SETPARM identifies the batch interface server by name. This should match the batch interface identifier specified in the appropriate queue manager's global system definition.

The MQPMQSC program can process multiple MQSeries commands, for example:

```
// JOB MQSCJOB
// SETPARM MQBISRV='MQBISRV2'
// LIBDEF *,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
// EXEC MQPMQSC,SIZE=AUTO
```

```
ALTER QLOCAL('ANYQ') GET(DISABLED)
DELETE CHANNEL('VSE1.TO.NT5')
DISPLAY QMGR CCSID
/*
/&
```

In addition, MQSeries commands can be split over multiple SYSIPT lines by using the plus character (+) to indicate continuation. For example:

```
// JOB MQSCJOB
// SETPARM MQBISRV='MQBISRV2'
// LIBDEF *,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
// EXEC MQPMQSC,SIZE=AUTO
ALTER CHANNEL('VSE1.TO.NT5') +
      CHLTYPE(SDR)           +
      CONVERT(NO)
/*
/&
```

MQSeries command prerequisites

There are several prerequisites that must be met before MQSeries commands can be processed by the MQPMQSC utility program. These include:

- MQSeries is installed and active in CICS.
  See Chapter 2, “Installation,” on page 7.
- System command queue is defined to the queue manager.
  See “Queue Manager Communications Settings” on page 69.
- System reply queue is defined to the queue manager.
  See “Queue Manager Communications Settings” on page 69.
- Batch interface is active in CICS.
  See “Using the batch interface” on page 136.
- PCF command server is active in CICS.
  See “Preparing MQSeries for PCF” on page 168.

Descriptions of the MQSeries commands

This section describes the MQSeries commands supported by MQSeries for VSE/ESA.

MQSeries commands can be divided into these categories:

- MQSeries channel commands
- MQSeries queue commands
- MQSeries queue manager commands

Note: In the following command descriptions, where a parameter requires an integer value, but a character string is provided, the value is interpreted as zero.

MQSeries channel commands

The MQSeries channel commands are:

ALTER CHANNEL
DEFINE CHANNEL
DELETE CHANNEL
DISPLAY CHANNEL
RESET CHANNEL
START CHANNEL

STOP CHANNEL

ALTER CHANNEL

Purpose: Use ALTER CHANNEL to alter the parameters of a channel.

Synonym:

```
ALT CHL
```

Syntax:

```
ALTER CHANNEL(channel-name) CHLTYPE(channel-type) optional-parameters
```

Parameters:

channel-name
Channel name. The channel-name value should match the name of a channel defined to the queue manager.

channel-type
Channel type. The channel-type value should match the channel type of the channel identified by channel-name. Valid channel types include:

SDR Sender.

RCVR Receiver.

SVRCONN
Server connection (used by clients).

optional-parameters
Optional parameters for the ALTER CHANNEL command include:

ALLOCRTY(integer)
APPC connection retry count.

ALLOCFST(integer)
APPC connection retry fast timer.

ALLOCSLW(integer)
APPC connection retry slow timer.

BATCHSZ(integer)
Batch size.

DESCR(string)
Description.

CONNAME(string)
Connection name.

CONVERT(NO/YES)
Whether sender should convert application data.

DISCINT(integer)
Disconnection interval.

DISCRTY(integer)
Disconnection retry count.

MAXMSGL(integer)
Maximum message length.

MSGDATA (string)
Message exit user data.

MSGEXIT (string)
Message exit name.

PORTNUM(integer)
TCP/IP port number.

RCVEXIT (string)
Receive exit name.

RCVDATA (string)
Receive exit user data.

SCYEXIT (string)
Security exit name.

SCYDATA (string)
Security exit user data.

SENDEXIT (string)
Send exit name.

SENDDATA (string)
Send exit user data.

SEQWRAP(integer)
Sequence number wrap.

SSLCIPH(string)
SSL cipher specification.

SSLCAUTH(REQUIRED/OPTIONAL)
SSL client authentication.

SSLPEER(string)
SSL peer name.

TPNAME(string)
Transaction program name.

TRPTYPE(LU62/TCP)
Transport (transmission protocol) type.

XMITQ(string)
Transmission queue name.

DEFINE CHANNEL

Purpose: Use DEFINE CHANNEL to define a new channel to the queue manager.

Synonym:

```
DEF CHL
```

Syntax:

```
DEFINE CHANNEL(channel-name) CHLTYPE(channel-type)
TRPTYPE(trptype) optional-parameters
```

Parameters:

channel-name
Channel name. The channel-name value should be unique; it should not match a channel name already defined to the queue manager.

channel-type
Channel type. The channel-type value should be the required channel type for the new channel. Valid types include:

SDR Sender.

RCVR Receiver.

SVRCONN
Server connection (used by clients).

trptype
Transport (transmission protocol) type.The trptype value should identify the required transport type for the new channel. Valid types include:

LU62 APPC LU 6.2 protocol.

TCP Transmission Control protocol.

optional-parameters
optional-parameters for the DEFINE CHANNEL command include:

ALLOCRTY(integer)
APPC connection retry count.

ALLOCFST(integer)
APPC connection retry fast timer.

ALLOCSLW(integer)
APPC connection retry slow timer.

BATCHSZ(integer)
Batch size.

CONNAME(string)
Connection name. The conname value should be the name of an LU 6.2 connection, or for TCP/IP sender channels, a remote hostname or IP address.

Unlike other MQSeries platforms, for TCP/IP channels, the connection name does not include a port number. For MQSeries for VSE/ESA, the port number is specified separately via the PORTNUM optional parameter.

DESCR(string)
Description.

CONVERT(NO/YES)
Whether sender should convert application data.

DISCINT(integer)
Disconnection interval.

DISCRTY(integer)
Disconnection retry count.

MAXMSGL(integer)
Maximum message length.

PORTNUM(integer)
TCP/IP port number.

MSGDATA (string)
Message exit user data.

MSGEXIT (string)
Message exit name.

RCVEXIT (string)
Receive exit name.

RCVDATA (string)
Receive exit user data.

SCYEXIT (string)
Security exit name.

SCYDATA (string)
Security exit user data.

SENDEXIT (string)
Send exit name.

SENDDATA (string)
Send exit user data.

SEQWRAP(integer)
Sequence number wrap.

SSLCIPH(string)
SSL cipher specification.

SSLCAUTH(REQUIRED/OPTIONAL)
SSL client authentication.

SSLPEER(string)
SSL peer name.

TPNAME(string)
Transaction program name.

XMITQ(string)
Transmission queue name.

DELETE CHANNEL

Purpose: Use DELETE CHANNEL to delete a channel definition.

Synonym:

```
DELETE CHL
```

Syntax:

```
DELETE CHANNEL(channel-name)
```

Parameters:

channel-name
Channel name. The channel-name value should match an existing channel defined to the queue manager.

DISPLAY CHANNEL

Purpose: Use DISPLAY CHANNEL to display a channel definition.

Synonym:

```
DIS CHL
```

Syntax:

```
DISPLAY CHANNEL(channel-name) requested-attributes
```

Parameters:

channel-name
Channel name. The channel-name value should match an existing channel defined to the queue manager.

requested-attributes
Attributes of the channel that are to be displayed. This can be:

ALL Displays all channel attributes. If you do not use ALL, you can use any combination of the other keywords.

ALTDATE
Last modification date.

ALTTIME
Last modification time.

ALLOCRTY
APPC connection retry count.

ALLOCFST
APPC connection retry fast timer.

ALLOCSLW
APPC connection retry slow timer.

BATCHSZ
Batch size.

DESCR
Channel description.

CHLTYPE
Channel type.

CONNAME
Connection name.

CONVERT
Whether sender should convert application data.

DISCINT
Disconnection interval.

DISCRTY
Disconnection retry count.

MAXMSGL
Maximum message length.

MSGEXIT
Message exit name.

MSGDATA
Message exit user data.

PORTNUM
TCP/IP port number.

RCVEXIT
Receive exit name.

RCVDATA
Receive exit user data.

SCYEXIT
Security exit name.

SCYDATA
Security exit user data.

SENDEXIT
Send exit name.

SENDDATA
Send exit user data.

SEQWRAP
Sequence number wrap.

SSLCIPH
SSL cipher specification.

SSLCAUTH
SSL client authentication.

SSLPEER
SSL peer name.

TPNAME
Transaction program name.

TRPTYPE
Transport (transmission protocol) type.

XMITQ
Transmission queue name.

RESET CHANNEL

Purpose: Use RESET CHANNEL to reset the message sequence number for an MQSeries channel with, optionally, a specified sequence number to be used the next time that the channel is started.

Synonym:

```
RESET CHL
```

Syntax:

```
RESET CHANNEL(channel-name) optional-parameter
```

Parameters:

channel-name
Channel name. The channel-name value should match an existing channel defined to the queue manager.

optional-parameter
The option parameter for RESET CHANNEL command is as follows:

SEQNUM(integer)
The new message sequence number, which must be greater than or equal to 1, and less than or equal to 999 999. If this parameter is not specified, the sequence number is reset to 1.

START CHANNEL

Purpose: Use START CHANNEL to start a channel.

Synonym:

```
STA CHL
```

Syntax:

```
START CHANNEL(channel-name)
```

Parameters:

channel-name
Channel name. The channel-name value should match an existing channel defined to the queue manager.

STOP CHANNEL

Purpose: Use STOP CHANNEL to stop a channel.

Synonym:

```
STOP CHL
```

Syntax:

```
STOP CHANNEL(channel-name)
```

Parameters:

channel-name
Channel name. The channel-name value should match an existing channel defined to the queue manager.

MQSeries queue commands

The MQSeries queue commands are:

- ALTER QALIAS
- ALTER QLOCAL
- ALTER QREMOTE
- DEFINE QALIAS
- DEFINE QLOCAL
- DEFINE QREMOTE
- DELETE QALIAS
- DELETE QLOCAL
- DELETE QREMOTE
- DISPLAY QALIAS
- DISPLAY QLOCAL
- DISPLAY QREMOTE

ALTER QALIAS

Purpose: Use ALTER QALIAS to alter the parameters of an alias queue.

Synonym:

```
ALT QA
```

Syntax:

```
ALTER QALIAS(q-name) optional-parameters
```

Parameters:

q-name
Queue name. The q-name value should specify an existing alias queue name defined to the queue manager.

optional-parameters
optional-parameters for the ALTER QALIAS command include:

DESCR(string)
Alias queue description.

GET(ENABLED/DISABLED)
Get uninhibit and inhibit.

PUT(ENABLED/DISABLED)
Put uninhibit and inhibit.

TARGQ(string)
Target queue of alias.

ALTER QLOCAL

Purpose: Use ALTER QLOCAL to alter the parameters of a local queue.

Synonym:

```
ALT QL
```

Syntax:

```
ALTER QLOCAL(q-name) optional-parameters
```

Parameters:

q-name
Queue name. The q-name value should specify an existing local queue name defined to the queue manager.

optional-parameters
optional-parameters for the ALTER QLOCAL command include:

DESCR(string)
Local queue description.

GET(ENABLED/DISABLED)
Get uninhibit and inhibit.

MAXDEPTH(integer)
Maximum queue depth.

MAXMSGL(integer)
Maximum message length.

MAXQUSER(integer)
Maximum number of active opens.

MAXGLOCK(integer)
Buffer size for queue manager to manage concurrent queue access.

MAXLLOCK(integer)
Buffer size for applications to manage concurrent queue access.

MAXTRIGS(integer)
Maximum number of concurrent trigger instances.

NOSHARE
Non-shareable queue. This parameter is mutually exclusive to the SHARE parameter.

NOTRIGGER
No trigger on queue. This parameter is mutually exclusive to the TRIGGER parameter.

NOTRIGREST
No trigger restart allowed. This parameter is mutually exclusive to the TRIGREST parameter.

PUT(ENABLED/DISABLED)
PUT uninhibit and inhibit.

QDEPTHHI(integer)
The threshold against which the queue depth is compare to generate a Queue Depth High event.

QDEPTHLO(integer)
The threshold against which the queue depth is compare to generate a Queue Depth Low event.

QDPHIEV(ENABLED/DISABLED)
Controls whether Queue Depth High events are generated.

QDPLOEV(ENABLED/DISABLED)
Controls whether Queue Depth Low events are generated.

QDPMAXEV(ENABLED/DISABLED)
Controls whether Queue Full events are generated.

QSVCIEV(HIGH/OK/NONE)
Controls whether Service Interval High or Service Interval OK events are generated.

QSVCINT(integer)
The service interval used for comparison to generate Service Interval High and Service Interval OK events.

SHARE
Shareable queue. This parameter is mutually exclusive to the NOSHARE parameter.

TRIGCHAN(string)
Channel name for MCA trigger process.

TRIGDATA(string)
User data passed to trigger instance.

TRIGGER
Trigger on queue. This parameter is mutually exclusive to the NOTRIGGER parameter.

TRIGPROG(string)
Program name for trigger process.

TRIGREST
Trigger restart allowed. This parameter is mutually exclusive to the NOTRIGREST parameter.

TRIGTERM(string)
Terminal identifier for trigger process.

TRIGTRAN(string)
Transaction identifier for trigger process.

TRIGTYPE(FIRST/EVERY)
Trigger type.

USAGE(NORMAL/XMITQ)
Queue usage.

ALTER QREMOTE

Purpose: Use ALTER QREMOTE to alter the parameters of a remote queue.

Synonym:

```
ALT QR
```

Syntax:

```
ALTER QREMOTE(q-name) optional-parameters
```

Parameters:

q-name
Queue name. The q-name value should specify an existing remote queue name defined to the queue manager.

optional-parameters
optional-parameters for the ALTER QREMOTE command include:

PUT(ENABLED/DISABLED)
Put inhibit and uninhibit.

RNAME(string)
Remote queue name.

RQMNAME(string)
Remote queue manager name.

XMITQ(string)
Transmission queue name.

DEFINE QALIAS

Purpose: Use DEFINE QALIAS to define a new alias queue, and set its parameters.

Note: An alias queue provides a level of indirection to another queue. The queue to which the alias refers must be another local or remote queue, defined at this queue manager. It cannot be another alias queue.

Synonym:

```
DEF QA
```

Syntax:

```
DEFINE QALIAS(q-name) optional-parameters
```

Parameters:

q-name
Queue name. The q-name value should specify a unique queue name that is not already defined to the queue manager.

optional-parameters
optional-parameters for the DEFINE QALIAS command include:

DESCR(string)
Alias queue description.

GET(ENABLED/DISABLED)
Get uninhibit and inhibit.

PUT(ENABLED/DISABLED)
Put uninhibit and inhibit.

TARGQ(string)
Target queue of alias.

DEFINE QLOCAL

Purpose: Use DEFINE QLOCAL to define a new local queue, and set its parameters.

Synonym:

```
DEF QL
```

Syntax:

```
DEFINE QLOCAL(q-name) CICSFILE(f-name) optional-parameters
```

Parameters:

q-name
Queue name. The q-name value should specify a unique queue name that is not already defined to the queue manager.

f-name
CICS file name for queue messages. The f-name value should specify a filename defined to the CICS region.

optional-parameters
optional-parameters for the DEFINE QLOCAL command include:

DESCR(string)
Local queue description.

GET(ENABLED/DISABLED)
Get uninhibit and inhibit.

MAXDEPTH(integer)
Maximum queue depth.

MAXMSGL(integer)
Maximum message length.

MAXQUSER(integer)
Maximum number of active opens.

MAXGLOCK(integer)
Buffer size for queue manager to manage concurrent queue access.

MAXLLOCK(integer)
Buffer size for applications to manage concurrent queue access.

MAXTRIGS(integer)
Maximum number of concurrent trigger instances.

NOSHARE
Non-shareable queue. This parameter is mutually exclusive to the SHARE parameter.

NOTRIGGER
No trigger on queue. This parameter is mutually exclusive to the TRIGGER parameter.

NOTRIGREST
No trigger restart allowed. This parameter is mutually exclusive to the TRIGREST parameter.

PUT(ENABLED/DISABLED)
Put uninhibit and inhibit.

QDEPTHHI(integer)
The threshold against which the queue depth is compare to generate a Queue Depth High event.

QDEPTHLO(integer)
The threshold against which the queue depth is compare to generate a Queue Depth Low event.

QDPHIEV(ENABLED/DISABLED)
Controls whether Queue Depth High events are generated.

QDPLOEV(ENABLED/DISABLED)
Controls whether Queue Depth Low events are generated.

QDPMAXEV(ENABLED/DISABLED)
Controls whether Queue Full events are generated.

QSVCIEV(HIGH/OK/NONE)
Controls whether Service Interval High or Service Interval OK events are generated.

QSVCINT(integer)
The service interval used for comparison to generate Service Interval High and Service Interval OK events.

SHARE
Shareable queue. This parameter is mutually exclusive to the NOSHARE parameter.

TRIGCHAN(string)
Channel name for MCA trigger process.

TRIGDATA(string)
User data passed to trigger instance.

TRIGGER
Trigger on queue. This parameter is mutually exclusive to the NOTRIGGER parameter.

TRIGPROG(string)
Program name for trigger process.

TRIGREST
Trigger restart allowed. This parameter is mutually exclusive to the NOTRIGREST parameter.

TRIGTERM(string)
Terminal identifier for trigger process.

TRIGTRAN(string)
Transaction identifier for trigger process.

TRIGTYPE(FIRST/EVERY)
Trigger type.

USAGE(NORMAL/XMITQ)
Queue usage.

DEFINE QREMOTE

Purpose: Use DEFINE QREMOTE to define a new remote queue, and set its parameters.

Synonym:

```
DEF QR
```

Syntax:

```
DEFINE QREMOTE(q-name) optional-parameters
```

Parameters:

q-name
Queue name. The q-name value should specify a unique queue name that is not already defined to the queue manager.

optional-parameters
optional-parameters for the DEFINE QREMOTE command include:

DESCR(string)
Remote queue description.

PUT(ENABLED/DISABLED)
Put uninhibit and inhibit.

RNAME(string)
Remote queue name.

RQMNAME(string)
Remote queue manager name.

XMITQ(string)
Transmission queue name.

DELETE QALIAS

Purpose: Use DELETE QALIAS to delete an alias queue definition.

Synonym:

```
DELETE QA
```

Syntax:

```
DELETE QALIAS(q-name)
```

Parameters:

q-name
Queue name. The q-name value should specify an existing alias queue name defined to the queue manager.

DELETE QLOCAL

Purpose: Use DELETE QLOCAL to delete a local queue definition.

Synonym:

```
DELETE QL
```

Syntax:

```
DELETE QLOCAL(q-name) optional-parameter
```

Parameters:

q-name
Queue name. The q-name value should specify an existing local queue name defined to the queue manager.

optional-parameter
The optional-parameters for the DELETE QLOCAL command is:

PURGE
Purge queue messages. If the queue contains messages and the PURGE parameter is not specified, the command will fail.

DELETE QREMOTE

Purpose: Use DELETE QREMOTE to delete a remote queue definition.

Synonym:

```
DELETE QR
```

Syntax:

```
DELETE QREMOTE(q-name)
```

Parameters:

q-name
Queue name. The q-name value should specify an existing remote queue name defined to the queue manager.

DISPLAY QALIAS

Purpose: Use DISPLAY QALIAS to display the attributes of an alias queue.

Synonym:

```
DIS QA
```

Syntax:

```
DISPLAY QALIAS(q-name) requested-attributes
```

Parameters:

q-name
Queue name. The q-name value should specify an existing alias queue name defined to the queue manager.

requested-attributes
Attributes of the alias queue that are to be displayed. This can be:

ALL Displays all alias queue attributes. If you do not use ALL, you can use any combination of the other keywords.

ALTDATE
Last modification date.

ALTTIME
Last modification time.

DESCR
Alias queue description.

GET GET inhibit and uninhibit.

PUT PUT inhibit and uninhibit.

TARGQ(string)
Target queue of alias.

DISPLAY QLOCAL

Purpose: Use DISPLAY QLOCAL to display the attributes of a local queue.

Synonym:

```
DIS QL
```

Syntax:

```
DISPLAY QLOCAL(q-name) requested-attributes
```

Parameters:

q-name
Queue name. The q-name value should specify an existing local queue name defined to the queue manager.

requested-attributes
Attributes of the local queue that are to be displayed. This can be:

ALL Displays all local queue attributes. If you do not use ALL, you can use any combination of the other keywords.

ALTDATE
Last modification date.

ALTTIME
Last modification time.

CICSFILE
CSD file name for queue messages.

DESCR
Local queue description.

GET Get inhibit and uninhibit.

MAXDEPTH
Maximum queue depth.

MAXMSGL
Maximum message length.

MAXQUSER
Maximum number of active opens.

MAXGLOCK
Buffer size for queue manager to manage concurrent queue access.

MAXLLOCK
Buffer size for applications to manage concurrent queue access.

MAXTRIGS
Maximum number of concurrent trigger instances.

NOSHARE
Non-shareable queue. This parameter is mutually exclusive to the SHARE parameter.

NOTRIGGER
No trigger on queue. This parameter is mutually exclusive to the TRIGGER parameter.

NOTRIGREST
No trigger restart allowed. This parameter is mutually exclusive to the TRIGREST parameter.

PUT PUT inhibit and uninhibit.

QDEPTHHI
The threshold against which the queue depth is compare to generate a Queue Depth High event.

QDEPTHLO
The threshold against which the queue depth is compare to generate a Queue Depth Low event.

QDPHIEV
Controls whether Queue Depth High events are generated.

QDPLOEV
Controls whether Queue Depth Low events are generated.

QDPMAXEV
Controls whether Queue Full events are generated.

QSVCIEV
Controls whether Service Interval High or Service Interval OK events are generated.

QSVCINT
The service interval used for comparison to generate Service Interval High and Service Interval OK events.

SHARE
Shareable queue. This parameter is mutually exclusive to the NOSHARE parameter.

TRIGCHAN
Channel name for MCA trigger process.

TRIGDATA
User data passed to trigger instance.

TRIGGER
Trigger on queue. This parameter is mutually exclusive to the NOTRIGGER parameter.

TRIGPROG
Program name for trigger process.

TRIGREST
Trigger restart allowed. This parameter is mutually exclusive to the NOTRIGREST parameter.

TRIGTERM
Terminal identifier for trigger process.

TRIGTRAN
Transaction identifier for trigger process.

TRIGTYPE
Trigger type.

USAGE
Queue usage.

DISPLAY QREMOTE

Purpose: Use DISPLAY QREMOTE to display the attributes of a remote queue.

Synonym:

```
DIS QR
```

Syntax:

```
DISPLAY QREMOTE(q-name) requested-attributes
```

Parameters:

q-name
Queue name. The q-name value should specify an existing remote queue name defined to the queue manager.

requested-attributes
Attributes of the remote queue that are to be displayed. This can be:

ALL Displays all remote queue attributes. If you do not use ALL, you can use any combination of the other keywords.

ALTDATE
Last modification date.

ALTTIME
Last modification time.

PUT Put inhibit and uninhibit.

RNAME
Remote queue name.

RQMNAME
Remote queue manager name.

XMITQ
Transmission queue name.

MQSeries queue manager commands

The MQSeries queue manager commands are:

- ALTER QMGR
- DISPLAY QMGR
- PING QMGR
- START LISTENER

ALTER QMGR

Purpose: Use ALTER QMGR to alter the queue manager parameters for the local queue manager.

Synonym:

```
ALT QMGR
```

Syntax:

```
ALTER QMGR qmgr-attrs
```

Parameters:

qmgr-attrs
Queue manager attributes for the ALTER QMGR command include the following:

AUTHOREV(ENABLED/DISABLED)
Whether authorization (Not Authorized) events are generated.

BATCHID(string)
Batch interface identifier.

BIAUTO
Automatic activation of the batch interface. This parameter is mutually exclusive of the NOBIAUTO parameter.

CCSID(integer)
Coded character set identifier.

COMMANDQ(string)
System command input queue.

CSAUTO
Automatic activation of the PCF command server. This parameter is mutually exclusive of the NOCSAUTO parameter.

CSCNVRT
Data conversion by command server of PCF messages. This parameter is mutually exclusive of the NOCSCNVRT parameter.

CSDLQ
Dead letter queue store by command server of undeliverable PCF reply messages. This parameter is mutually exclusive of the NOCSDLQ parameter.

DEADQ(string)
Dead letter queue name.

DESCR(string)
Queue manager description.

INHIBTEV(ENABLED/DISABLED)
Whether inhibit (Inhibit Get and Inhibit Put) events are generated.

LISTPORT(integer)
TCP/IP listener port number.

LOCALEV(ENABLED/DISABLED)
Whether local error events are generated.

LOGQ(string)
System log queue name.

MAXDEPTH(integer)
Maximum queue depth for local queues.

MAXGLOCK(integer)
Buffer size for queue manager to manage concurrent queue access.

MAXHANDS(integer)
Maximum number of connections to queue manager.

MAXLLOCK(integer)
Buffer size for applications to manage concurrent queue access.

MAXMSGL(integer)
Maximum length of messages for local queues.

MAXQOPEN(integer)
Maximum number of concurrently open queues.

MAXQUSER(integer)
Maximum number of active opens to any particular queue.

MONINTVL(integer)
Queue manager housekeeping process interval.

MONITORQ(string)
MQI diagnostic queue name.

NOBIAUTO
Non-automatic activation of the batch interface. This parameter is mutually exclusive of the BIAUTO parameter.

NOCSAUTO
Non-automatic activation of the PCF command server. This parameter is mutually exclusive of the CSAUTO parameter.

NOCSCNVRT
No data conversion by command server of PCF messages. This parameter is mutually exclusive of the CSCNVRT parameter.

NOCSDLQ
No dead letter queue store by command server of undeliverable PCF reply messages. This parameter is mutually exclusive of the CSDLQ parameter.

PERFMEV(ENABLED/DISABLED)
Whether performance-related events are generated.

REMOTEEV(ENABLED/DISABLED)
Whether remote error events are generated.

REPLYQ(string)
System command reply queue name for MQSC processing.

SSLKEYL(string)
SSL key library name.

SSLKEYM(string)
SSL key library member name.

STRSTPEV(ENABLED/DISABLED)
Whether start and stop events are generated.

DISPLAY QMGR

Purpose: Use DISPLAY QMGR to display the attributes of the queue manager.

Synonym:

```
DIS QMGR
```

Syntax:

```
DISPLAY QMGR requested-attributes
```

Parameters:

requested-attributes
Attributes of the queue manager that are to be displayed. This can be:

ALL Displays all queue manager attributes. If you do not use ALL, you can use any combination of the other keywords.

ALTDATE
Last modification date.

ALTTIME
Last modification time.

AUTHOREV
Whether authorization (Not Authorized) events are generated.

BATCHID
Batch interface identifier.

BIAUTO
Automatic activation of the batch interface. This parameter is mutually exclusive of the NOBIAUTO parameter.

CCSID
Coded character set identifier.

COMMANDQ
System command input queue.

CSAUTO
Automatic activation of the PCF command server. This parameter is mutually exclusive of the NOCSAUTO parameter.

CSCNVRT
Data conversion by command server of PCF messages. This parameter is mutually exclusive of the NOCSCNVRT parameter.

CSDLQ
Dead letter queue store by command server of undeliverable PCF reply messages. This parameter is mutually exclusive of the NOCSDLQ parameter.

DEADQ
Dead letter queue name.

DESCR
Queue manager description.

INHIBTEV
Whether inhibit (Inhibit Get and Inhibit Put) events are generated.

LISTPORT
TCP/IP listener port number.

LOCALEV
Whether local error events are generated.

LOGQ
System log queue name.

MAXDEPTH
Maximum queue depth for local queues.

MAXGLOCK
Buffer size for queue manager to manage concurrent queue access.

MAXHANDS
Maximum number of connections to queue manager.

MAXLLOCK
Buffer size for applications to manage concurrent queue access.

MAXMSGL
Maximum length of messages for local queues.

MAXQOPEN
Maximum number of concurrently open queues.

MAXQUSER
Maximum number of active opens to any particular queue.

MONINTVL
Queue manager housekeeping process interval.

MONITORQ
MQI diagnostic queue name.

NOBIAUTO
Non-automatic activation of the batch interface. This parameter is mutually exclusive of the BIAUTO parameter.

NOCSAUTO
Non-automatic activation of the PCF command server. This parameter is mutually exclusive of the CSAUTO parameter.

NOCSCNVRT
No data conversion by command server of PCF messages. This parameter is mutually exclusive of the CSCNVRT parameter.

NOCSDLQ
No dead letter queue store by command server of undeliverable PCF reply messages. This parameter is mutually exclusive of the CSDLQ parameter.

PERFMEV
Whether performance-related events are generated.

REMOTEEV
Whether remote error events are generated.

REPLYQ
System command reply queue name for MQSC processing.

SSLKEYL
SSL key library name.

SSLKEYM
SSL key library member name.

STRSTPEV
Whether start and stop events are generated.

PING QMGR

Purpose: Use PING QMGR to test whether the queue manager is responsive to commands.

Synonym:

```
PING QMGR
```

Syntax:

```
PING QMGR
```

START LISTENER

Purpose: Use START LISTENER to start a TCP/IP listener task.

Synonym:

```
STA LSTR
```

Syntax:

```
START LISTENER optional-parameter
```

Parameters:

optional-parameter
The START LISTENER command can include the following optional parameter:

TRPTYPE(TCP/LU62)
Transport (transmission protocol) type. If this parameter is not specified, the default is TCP. If LU62 is specified, the command is successful, but does nothing.

Chapter 10. Secure Sockets Layer services

Secure Sockets Layer (SSL) is a communications protocol that provides secure communications over an open communications network (for example, the Internet). The SSL protocol is a layered protocol that is intended to be used on top of a reliable transport, such as Transmission Control Protocol (TCP/IP). SSL provides data privacy and integrity as well as server and client authentication based on public key certificates. Once an SSL connection is established between a client and server, data communications between client and server are transparent to the encryption and integrity added by the SSL protocol.

MQSeries for VSE/ESA incorporates SSL services between itself and remote queue managers and MQ clients that also incorporate SSL services. From an SSL perspective, in every case, the initiating application is considered the client and the remote application accepting the connection, the server. From an MQSeries perspective, the client is a remote sender Message Channel Agent (MCA) or an MQSeries client, and the server is the receiver MCA.

MQSeries activates SSL services on a per channel basis. This is possible through the channel definition. Each channel can identify whether or not SSL services are required when a connection is made or accepted, to or from a remote system or client program.

There are a number of steps involved in establishing an SSL enabled and active channel. These include:
- Installing the SSL feature
- Configuring the queue manager for SSL
- Configuring a channel for SSL
- Activating SSL services

Each of these is described in some detail below.

Installing the SSL feature

Before SSL channels can be established by MQSeries, the SSL feature must be installed and available under the VSE/ESA environment.

SSL for VSE is an optional product that is integrated into TCP/IP for VSE. As an optional product, it requires a special product code to activate its features. Full details for installation should be found in SSL for VSE documentation.

Part of the installation process, of immediate relevance to MQSeries, is the creation of the SSL key-ring sublibrary. The SSL key-ring sublibrary contains private key members (.PRVK files) and X.509 certificate files (.CERT files).

If SSL enabled channels are required, MQSeries for VSE/ESA must be configured with the name of the SSL key-ring sublibrary, and the name of the queue manager's private key and certificate files. This configuration is part of the queue manager definition.

© Copyright IBM Corp. 1995, 2004

Configuring the queue manager for SSL

Once the SSL feature has been installed and is available under VSE/ESA, the MQSeries queue manager can be configured to identify the SSL key-ring sublibrary and private key and certificate files.

Access SSL configuration for the queue manager from the Global System Definition maintenance screen (MQMT option 1.1), using PF9. Pressing PF9 displays the queue manager "Communication Settings" (figure 60).

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
14:47:42                   Global System Definition                  CIC1
MQMMSYS                     Communications Settings                  A000

    TCP/IP settings                          Batch Interface settings
    TCP/IP listener port : 01415             Batch Int. identifier: MQBISRV2
    Licensed clients . . : 00000             Batch Int. auto-start: Y
    Adopt MCA  . . . . . : Y
    Adopt MCA Check  . . : Y

    SSL parameters
    Key-ring sublibrary  : PRD2.SSLKEYS
    Key-ring member  . . : VSE1CER

    PCF parameters
    System command queue : SYSTEM.ADMIN.COMMAND.QUEUE
    System reply queue . : SYSTEM.ADMIN.REPLY.QUEUE
    Cmd Server auto-start: Y
    Cmd Server convert . : N
    Cmd Server DLQ store : Y

Requested record displayed.
PF2=Queue Manager details  PF3=Quit   PF4/Enter=Read   PF6=Update
```

Figure 60. Queue manager communication settings

Queue manager communication settings are divided into four categories:
- TCP/IP settings
- SSL parameters
- PCF parameters
- Batch interface settings

Of immediate relevance to SSL enabled channels are the TCP/IP settings and the SSL parameters.

TCP/IP settings

TCP/IP listener port

The TCP/IP listener port represents the IP port number on which the MQ Listener program will accept remote connection requests. The MQ Listener is a long running transaction, MQTL.

Caution should be taken not to use a port number that is already in use by another application or subsystem. The default value for MQSeries is 1414.

Licensed clients

The number of licensed clients represents the maximum number of concurrent client connections that will be accepted by the queue manager. The number allowed is determined by your MQSeries for VSE/ESA license agreement.

Current license agreements do not restrict the number of concurrent clients. However, the license agreement should be checked for restrictions before setting this field to a value other than zero.

SSL parameters

Key-ring sublibrary

The key-ring sublibrary identifies the SSL key-ring sublibrary, identified and generated during SSL for VSE installation. The key- ring sublibrary contains private key and X.509 certificate files. The value entered should be a valid VSE sublibrary name.

If a key-ring sublibrary is specified, MQSeries will perform SSL initialization during system startup, even if there are no SSL enabled channels. If SSL is not installed, this field should be left blank.

Key-ring member

The key-ring member identifies the SSL key-ring sublibrary member name of the private key and certificate files that will be used by MQSeries enabled channels. This must be a valid VSE sublibrary member name.

It should be noted that MQSeries for VSE/ESA uses the same private key and certificate for all SSL enabled channels. It is not possible to identify a different certificate on a per channel basis. Consequently, the key-ring member name should identify a private key and certificate files appropriate for all SSL enabled channels.

Configuring a channel for SSL

Sender, receiver and client channels can be enabled for SSL. In the case of Sender and Receiver channels, SSL enablement assumes that the partner channel definition (on a remote MQ system) is also configured for SSL. For client channels, MQSeries documentation for the relevant client system should be reviewed to determine how to enable a client for SSL.

Configure the SSL parameters for a channel using the “Channel SSL Parameters” screen. To get to this screen, press PF10 at the “Maintain Channel Record” screen (MQMT option 1.3)

```
11/28/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
12:44:28                  Channel SSL Parameters                    CIC1
MQMMCHN                                                             A003

     Channel Name: VSE1.TCP.NT1           Type: S

     SSL Cipher Specification. : 02       (2 character code)
     SSL Client Authentication : R        (Required or Optional)

     SSL Peer Attributes:
     > O=IBM,OU="Australian Programming Centre",C=Australia,ST=WA,L=Per <
     > th,CN=www.ibm.com                                               <
     >                                                                 <
     >                                                                 <









SSL channel parameters displayed.
    PF2 = Return    PF3 = Quit     PF4 = Read      PF6 = Update
```

Figure 61. SSL parameters for a channel

SSL parameters are identical, regardless of channel type. Consequently, the above screen is applicable to sender, receiver and client channels.

Secure Sockets Layer services are only available for TCP/IP channels. Consequently, SSL parameters for SNA LU 6.2 channels are ignored.

SSL channel parameters

Channel name
: The name of the channel for which the SSL parameters apply. This is a display field only.

Type
: The type of the channel for which the SSL parameters apply. This is a display field only.

SSL Cipher Specification
: The SSL Cipher Specification is a two-character code that identifies an SSL version 3 cipher specification supported by the SSL for VSE feature. For example:

Table 10. Supported SSL cipher specifications

Cipher	Description
01	NULL MD5
02	NULL SHA
08	DES40 SHA for Export
09	DES SHA for U.S.
0A	Triple DES SHA for U.S.
62	RSA_EXPORT1024_DESCBC_SHA

The code selected must be supported on the remote system. In the case of a sender channel, MQSeries for VSE/ESA will establish an SSL enabled channel

with a remote MQ system only if the remote system accepts the specified code. For receiver channels, the remote system will identified the desired code. If the local SSL feature supports the designated code, channel initialization will proceed. Otherwise the channel is terminated with an error.

It should be noted that this parameter determines whether or not the channel is SSL enabled. If this field is blank, the other SSL parameters are ignored, and the channel operates without SSL services. Any non-blank value means the channel is SSL enabled, and the other SSL parameters are used during channel initialization.

SSL Client Authentication

The SSL Client Authentication field can be set to 'R' for required, or 'O' for optional. If client authentication is required, MQSeries checks that a certificate was sent from the remote system during SSL initial negotiation. If not, the channel is terminated with a error.

Since a certificate is always sent by the receiver (or SSL server) to the sender (or SSL client), this field is only meaningful to receiver and client channels. However, the MQSeries for VSE/ESA Receiver MCA, which acts as the SSL server, requires that SSL clients send a certificate during SSL negotiation. Consequently, under MQSeries for VSE/ESA, a client certificate will always be received during SSL negotiation. If not, the channel is terminated with an initialization error.

The SSL client authentication field, therefore, exists for compatibility with other MQSeries systems and possible future expansion. To make this apparent, this field should be set to 'R'.

SSL Peer Attributes

The SSL Peer Attributes parameter allows a channel to verify that the partner's certificate contains certain identifiable characteristics. If the partner's certificate does not contain these characteristics, the channel is terminated with an error.

Identifiable characteristic types include the following:

Table 11. SSL Peer Attribute types

Type	**Description**
CN	Common name
L	Locality
ST	State or province
C	Country
O	Organization
OU	Organizational unit
SERIAL	Serial number

The characteristic types pertain to the Subject attributes of the X.509 certificate, except for serial number, which pertains to the Serial Number of the certificate.

The SSL Peer Attributes field takes the following form:

type=value,type=value,etc. Where *type* is one of the characteristic types listed in Table 11, the equals sign (=) is constant, and *value* identifies an expected value relevant to the characteristic type specified. For multiple attributes, the comma (,) is also required and constant. For example:

```
O=IBM,C=US
```

In this example, the remote partner's certificate must have a Subject Organization of ″IBM″, and a Subject Country of ″US″.

The SSL Peer Attributes parameter will also accept wildcards (*). Each value can have only one wildcard. Wildcards cannot be imbedded in a value. For example:

```
O=IBM,OU=LAB*,C=UK
```

In this example, the channel will accept remote certificates with a Subject Organization of ″IBM″, a Subject Country of ″UK″ and any Subject Organizational Unit beginning with ″LAB″.

The SSL Peer Attributes parameter will also accept imbedded spaces. These must be enclosed in double quotes (″). For example:

```
O="IBM GSA"
```

Double quotes are optional for values that do not contain imbedded blanks.

If the SSL Peer Attribute parameter is set and the remote certificate does not match its stipulations, the channel is terminated with an error. If the SSL Peer Attributes is not set (it is left blank), the remote certificate's identifying attributes are not examined. In other words, a blank parameter means the channel can be activated by any valid certificate.

Activating SSL services

When a channel is configured for SSL, MQSeries automatically activates SSL services when the channel is established.

For Sender channels, the Sender Message Channel Agent (MCA), trigger program MQPSEND, examines the revelant channel definition to check if the SSL Cipher Specification field has been set. If so, MQSeries considers the channel ″SSL enabled″ and will attempt to establish an SSL connection with the remote queue manager using SSL services.

For Receiver and Client channels, the Receiver MCA does not, on invocation, have details of the channel. Consequently, it cannot examine the SSL Cipher Specification parameter. Instead, the Receiver MCA examines the initial dataflow from the remote queue manager. If it is identifiable as an SSL exchange, the Receiver MCA will attempt to secure the connection using SSL services. If successful, subsequent dataflow will identify the appropriate channel.

It is at this point that MQSeries will verify that the correct SSL Cipher Specification, SSL Client Authentication and SSL Peer Attributes have been received or negotiated. If not, the channel is immediately terminated with an error.

This means that for all SSL enabled channels, the dataflow, from start to finish, is under SSL control. This included MQSeries' initial channel neogiation which contains channel, queue manager and target queue information.

The SSL Peer Attributes parameter adds further protection by allowing a channel to reject any connection that uses a certificate that does not meet the stipulations of the parameter.

Chapter 11. Security

This chapter describes the features of security control in MQSeries for VSE/ESA and how you can implement and manage this control.

Note: Examples in this chapter uses CA-Top Secret® as an external security manager (ESM). If you are using a different ESM, you should modify the techniques described.

Where profile names are shown, replace the subsystem identifier (ssid) in the profile name with the name of the MQSeries subsystem you are using. As a general rule, you can use your queue manager name as your subsystem identifier.

Why you need to protect MQSeries resources

Because MQSeries handles the transfer of information that is potentially valuable, you need the safeguard of a security system. This ensures that the resources that MQSeries owns and manages are protected from unauthorized access, which could lead to the loss or disclosure of the information. In a secure system, it is essential that none of the following are accessed or changed by any unauthorized user or process:

- Connections to MQSeries
- MQSeries objects such as queue managers and queues
- MQSeries transmission links
- MQSeries system control commands
- MQSeries messages
- Context information associated with messages

To provide the necessary security, MQSeries uses the VSE system authorization facility (SAF) to route authorization requests to an ESM, for example, CA-Top Secret.

The decision to allow access to an object is made by the ESM, and MQSeries follows that decision. If the ESM cannot make a decision, MQSeries prevents access to the object by default. However, by default, if the CICS system running MQSeries is configured without security, MQSeries will not restrict access to its resources.

Implementing MQSeries security

It is easier to set up and administer your security if first you decide on a set of naming conventions for your MQSeries objects.

To implement a security strategy for your MQSeries subsystem, you must decide:

- How security is to be used and implemented.
- Who is going to use the MQSeries system and resources.

To use the CA-Top Secret examples, as shown in this manual, you must be a suitably authorized user, for example, the MSCA user. You can enter the commands either from CICS or via a batch job, using the TSS transaction or the TSSCMNDB program, respectively.

© Copyright IBM Corp. 1995, 2004

Resources you can protect

When MQSeries starts, or when it is instructed by an operator command, MQSeries determines which resources you want to protect. You can control which security checks are performed for each individual queue manager. For example, you could implement a number of security checks on a production queue manager, but none on a test queue manager.

Objects protected by MQSeries for VSE/ESA include:

- connections
- queues
- messages
- commands
- command resources

This chapter also explains how you might protect MQSeries datasets, specifically, the VSAM files used by MQSeries.

Connection security

Connection security checking occurs either when an application program tries to connect to a queue manager by issuing an MQCONN request, or when MQSeries itself issues a connection request. You can turn connection security checking off for a particular MQSeries subsystem, but if you do, any user can connect to that subsystem.

MQSeries itself issues a connection request when it attempts to log messages to the system log. The logging mechanism writes messages to a transient data queue (MQER) that triggers a transaction to write the message to the system log queue. This transaction (MQER) runs as the CICS default user, or a user specified in the DCT entry for the transient data queue. At installation, you must decide whether to use the default user or a specific user to connect and write messages to the system log.

Similarly, the message expiry feature of MQSeries uses a transient data queue. The message expiry transient data queue (MQXP) is defined in the CICS Destination Control Table (DCT) to fire a trigger transaction (also MQXP) when the data queue contains at least one entry. The MQXP transaction is responsible for clearing expired messages from application queues, and for generating expiry report message when they are required.

Like the MQER transaction, the MQXP transaction runs with the authority of the CICS default user unless the DCT definition includes the USERID parameter. See “Changing the MQXP TDQ definition” on page 14. Since the MQXP transaction may need to place a report message on any application queue, the user that runs the transaction must have suitable authority. The user will at least require connect authority.

Queue and message security

Resources are checked when an application opens an object with an MQOPEN or an MQPUT1 call. The access needed to open an object depends on which open options are specified when the queue is opened.

A security check is performed when the queue manager object is opened. In this situation, the queue manager is protected in the same way as a queue object, that is, a user must have permission to access `ssid.qmname`, where `qmname` is the name of your queue manager.

Queue security controls who is allowed to open which queue, and what options they are allowed to open it with. For example, a user might be allowed to open a queue called PAYROLL.INCREASE.SALARY to browse the messages on the queue (via the MQOO_BROWSE option), but not to remove messages from the queue (via one of the MQOO_INPUT_* options). If you turn checking for queues off, any user can open any queue with any valid open option (that is, any valid MQOO_* option on an MQOPEN or MQPUT1 call).

Command security

Command security involves protecting the creation, deletion, and modification of MQSeries objects. Commands that affect MQSeries objects can be issued in three ways:

- Master terminal commands
- PCF commands
- MQSeries commands

Master terminal commands are those entered interactively in native CICS, or via the CICS Web Support feature from a web browser. Master terminal commands are generally selected from the primary options menu of the master terminal transaction (MQMT).

PCF commands are processed as data messages by the MQSeries command server which is a long-running CICS transaction (MQCS). The command server reads PCF messages from system command queue. The name of this queue is configurable as a communication parameter of the queue manager's global system definition. Message read from the system command queue are expected to be PCF commands which are parsed and processed by the MQSeries command processor (transaction MQCX). For more information about PCF commands, refer to Chapter 8, "Programmable system management," on page 167.

MQSeries commands are verb-based text messages processed by the MQSeries MQSC batch utility program (MQPMQSC). The MQSC utility program converts SYSIPT verb-based command text into PCF Escape messages and places them on the system command queue. Responses to these commands are placed on the system reply queue (a communication parameter of the global system definition). The MQSC utility processes MQSC reply messages and generates SYSLST text output. For more information about MQSeries commands and the command utility, refer to Chapter 9, "MQSeries commands," on page 253.

Command security involves authorization by command type. For example, a user might be authorized to issue 'DISPLAY' commands, but not 'ALTER' commands. Similarly, a user may be authorized to perform the display options but not the maintenance options under MQMT option 1 (Configuration).

Command resource security

Command resource security involves protecting MQSeries objects by name, and works in conjunction with command security. For example, a user may be authorized to issue 'DISPLAY' commands, but may be restricted to displaying objects with a certain prefix.

Consequently, for a user to display the details of a specific queue, for example, that user would need command authorization to 'DISPLAY' and command resource authorization to display details of the specific queue.

Dataset security

MQSeries for VSE/ESA queues are implemented as VSAM KSDS datasets, and MQSeries configuration is also stored in VSAM datasets. Therefore, it is important that these datasets are protected against unauthorized access under VSE generally.

Check your ESM documentation for specific details on protecting datasets. MQSeries assumes that users with authorization to specific queues, with specific access permissions, are also authorized to the datasets that contain queue data, with the same permissions. This assumption relies on the security administrator correlating the correct permissions for queues and datasets.

See Appendix H, "Security implementation," on page 499 for more details on how to protect your datasets.

Using security classes and resources

CA-Top Secret classes are used to hold the resources required for MQSeries security checking. Each class holds one or more resources used at some point in the checking sequence.

Table 12. Classes used by MQSeries

Member class	Description
MQADMIN	Used mainly for holding resources for administration-type functions. For example, profiles for MQSeries security switches.
MQCONN	Profiles used for connection security.
MQQUEUE	Profiles used in queue resource security.
MQCMDS	Profiles for command security.

Depending on your External Security Manager, these classes may be predefined. For CA-Top Secret, these are predefined Prefixed resources. To activate such resources, ensure that the the following setting exists in your CA-Top Secret parameter file:

```
FACILITY(CICSPROD=RES)
```

Notes:

1. CICSPROD should be replaced by the facility you are using for your MQSeries CICS region, if it is different.
2. After you change the parameter file, you need to restart your ESM.

Resources

All resources used by MQSeries are prefixed with the name of the subsystem that they are to be used by. For example, if queue manager VSE.QM1 has a queue called QUEUE_FOR_LOST_CARD_LIST, the appropriate profile would be defined to the ESM in class MQQUEUE as:

VSE.QM1.QUEUE_FOR_LOST_CARD_LIST

This means that different MQSeries subsystems sharing the same ESM database can have different security options. The subsystem identifier for the resource cannot be generic.

Switch resources

To control the security checking performed by MQSeries, you must define switch profiles. A switch profile is a normal resource that has a special meaning to MQSeries. If you do not want to control security checking, that is, allow MQSeries to check authority for all MQSeries resources, you do not need to define switch profiles.

Each 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 the queue manager is running, the changes will not be recognized until MQSeries is stopped and the MQSeries environment is re-established by the MQSE transaction.

The switch resources must always be defined in the MQADMIN class. The following table shows the valid switch profiles and the security type they control.

Note: In the descriptions that follow, the part of each resource name shown in upper case must be entered exactly as shown. The lower case 'ssid' part must be replaced by the queue manager name for the MQSeries subsystem you are setting up.

Table 13. Switch Resources

Switch Resource Name	Description
ssid.NO.SUBSYS.SECURITY	Subsystem security
ssid.NO.CONNECT.CHECKS	Connection security
ssid.NO.QUEUE.CHECKS	Queue security
ssid.NO.CMD.CHECKS	Command security
ssid.NO.CMD.RESC.CHECKS	Command resource security

If you intended to use the security switches, you can create them and grant access as follows:

```
TSS ADD(mqowner) MQADMIN(ssid.NO.CONNECT.CHECKS)
TSS PER(mqstart) MQADMIN(ssid.NO.CONNECT.CHECKS) ACC(READ)
```

In this example, the resource is owned by user `mqowner`, and the `mqstart` user is granted read access to the resource. Note that access to security switch resources is only relevant to the MQSeries for VSE/ESA startup user (that is, the user who starts MQSeries for VSE/ESA using MQSE, MQIT, or MQMT).

In the preceding example, security checks for connecting to the MQSeries for VSE/ESA queue manager would be disabled.

How switches work

MQSeries maintains an internal set of switches, which is associated with each of the switch resources shown in Table 13. 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 that switch are bypassed.

When a queue manager is started, first it checks the status of the resource switches. The queue manager sets its subsystem security switch off only if the switch resources exist and are readable by the user associated with MQSeries startup. In all other situations, the switches are set on. Note that switches are only applicable when MQSeries is installed with security active.

If the ssid.NO.SUBSYS.SECURITY resource is detected during startup, connection, queue, and message security is bypassed, regardless of other switch settings. This means it is possible to completely disable MQSeries object security by creating the NO.SUBSYS.SECURITY resource, making it readable to the startup user, and restarting MQSeries.

Take care with generic resources. Some ESMs automatically grant access to resources if the prefix of the resource is owned, or accessible to, a user. For example, if the resource ssid is created and owned by user MQM, and that resource is generic, some ESMs may automatically grant read access to ssid.* to user MQM. The result is that when MQSeries is started up by user MQM, MQSeries will assume all of the switches exist, and all object security will be disabled.

Protecting MQSeries resources

As well as optionally defining switch resources, ESM resources must be defined to protect the MQSeries objects.

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 switch profiles that you have deactivated.

Resource definitions 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 READ access to the appropriate profile.

Resource names for checking connections to MQSeries for VSE/ESA take the form:

ssid.CICS

This applies to CICS applications, batch programs using the batch interface, and remote clients. This is because all connections to MQSeries for VSE/ESA are effectively maintained within CICS.

For example, to grant user JOHNS connection authority to queue manager VSE.QM1, you must first define the resource and grant ownership:

```
TSS ADD(MQOWNER) MQCONN(VSE.QM1.CICS)
```

You can then grant connection authority as follows:

```
TSS PER(JOHNS) MQCONN(VSE.QM1.CICS) ACC(READ)
```

Depending on your ESM, the owner of a resource may by default have full authority. This would mean that user MQOWNER, in this example, would automatically be granted connection authority to queue manager VSE.QM1.

Batch connections

Security for batch connections is a special case. Batch programs connect to MQSeries for VSE/ESA running under CICS via the MQSeries for VSE/ESA batch interface.

Programs executed from a batch partition should use the // ID statement to identify their user and password. Security for batch programs should be established to verify the user and password identified on the // ID card.

A sample batch job might appear as follows:

```
// JOB MQBATCH
// ID USER=JOHNS,PWD=JOHNSPWD
// EXEC MYMQPROG
/*
/&
```

The MQ/VSE batch interface uses the user name identified in the // ID card and passes it to an interface transaction running under CICS. The interface transaction must be started by, and running as, a user identified to your ESM as a SURROGATE for the user identified on the // ID card.

To identify a user as a surrogate for another, you can use a command similar to:

```
TSS ADD(MQBATCH) SURROGAT(JOHNS)
```

where MQBATCH is the user that starts the batch interface transaction (MQBI) in CICS.

When the MQSeries batch program attempts to connect to the queue manager, a check for the surrogate rights of the interface user is issued. If this is successful, a partner transaction (MQBX) is started as the user identified on the // ID card. Therefore, the user identified on the // ID card should be known to CICS.

Once the partner transaction is started, it functions on behalf of the MQSeries batch program. This means that all MQI calls are executed as the user identified on the // ID card. For connection security, this user must be granted READ access to the ssid.CICS resource.

Client connections

Security for client connections is also a special case. For client connections, the client program runs on a remote system. Security for the execution of such programs remains the responsibility of the remote system.

For client programs, the MQSeries for VSE/ESA server program effectively performs MQSeries API requests on behalf of the client program. The server program runs under CICS and is executed as the MQSeries for VSE/ESA startup user. The startup user is the user who starts MQSeries for VSE/ESA using the MQSE, MQIT or MQMT transactions.

The MQSeries for VSE/ESA server program identifies the client user when the client connection is initiated with the MQCONN call. For authentication, the environment of the client program must include the MQ_USER_ID and MQ_PASSWORD environment variables. The values of these variables are passed

to the MQSeries for VSE/ESA server program when the connection begins. These variables should contain a valid user id and password, respectively, that are known to the VSE ESM.

The MQSeries for VSE/ESA server program, having identified and verified the client user and password, then performs all security checks for that user, not the MQSeries for VSE/ESA startup user.

This means that the client user must have the appropriate access to the required ESM resources. This is the same access that would be required for a normal CICS transaction user.

For example, for a client program that identifies itself as user JANED, and intends to connect to MQSeries for VSE/ESA and browse queue EMPLOYEE.DETAILS on VSE queue manager VSE.QM1, you would need to define and grant access to the following resources:

```
TSS PER(JANED) MQCONN(VSE.QM1.CICS) ACC(READ)
TSS PER(JANED) MQQUEUE(VSE.QM1.EMPLOYEE.DETAILS) ACC(READ)
```

Because authentication is possible only for client programs that identify themselves using the MQ_USER_ID and MQ_PASSWORD environment variables, MQSeries for VSE/ESA security for client programs is possible only for remote systems that support this protocol.

Another consideration, which may affect Java program clients, is access permission to the queue manager object. Some existing MQSeries Java classes open the queue manager object when they establish an initial connection. This means that users using MQSeries Java classes should be granted READ access to the MQSeries queue manager object.

For example:

```
TSS PER(cliuser) MQQUEUE(ssid.ssid) ACC(READ)
```

Resource definitions for queue security

If queue security is active, you must define resources in the MQQUEUE class, and permit the necessary groups or user IDs access to these resources, so that they can issue MQSeries API requests that use queues.

Resource names for queue security take the form:

ssid.queuename

where queuename is the name of the queue being opened, as specified in the object descriptor on the MQOPEN or MQPUT1 call. It may also be the name of the queue manager.

The ESM access required to open a queue depends on the MQOPEN or MQPUT1 options specified. If more than one of the MQOO_* options is coded, the queue security check is performed for the highest ESM authority required.

Table 14. Access levels for queue security

MQOPEN or MQPUT1 option	**ESM access level required to access ssid.queuename**
MQOO_BROWSE	READ
MQOO_INQUIRE	READ

Table 14. Access levels for queue security (continued)

MQOPEN or MQPUT1 option	ESM access level required to access ssid.queuename
MQOO_INPUT_*	UPDATE
MQOO_OUTPUT or MQPUT1	UPDATE
MQOO_SET	ALTER

For example, to grant user JOHNS authority to browse queue PAY.LIST on queue manager VSE.QM1:

```
TSS ADD(MQOWNER) MQQUEUE(VSE.QM1.PAY.LIST)
TSS PER(JOHNS) MQQUEUE(VSE.QM1.PAY.LIST) ACC(READ)
```

Alternatively, to grant user JOHNS authority to get and put messages to queue PAY.LIST on VSE.QM1:

```
TSS PER(JOHNS) MQQUEUE(VSE.QM1.PAY.LIST) ACC(READ,UPDATE)
```

Note that the resource only needs to be created, and ownership applied, once. Therefore, the TSS ADD command is issued only once for each queue resource defined to class MQQUEUE.

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 whether 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, a user only needs authorization to access the queue PAYROLL.REQUEST. There is no check whether that user is authorized to access the queue PAY.REQUEST.

Using alias queues with MQGET and MQPUT

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. You can protect a queue by defining two aliases that resolve to that queue:

- One that enables applications to get message from the queue.
- One that enables applications to put messages on the queue.

The following text is an example of defining your queue to MQSeries (these definitions are based on OS/2 formats, and you should use the MQSeries for VSE/ESA Master Terminal transaction to create appropriate definitions):

```
DEFINE QLOCAL(USE_ALIAS_TO_ACCESS) GET(ENABLED)
       PUT(ENABLED)

DEFINE QALIAS(USE_FOR_GETS) GET(ENABLED)
       PUT(DISABLED) TARGQ(USE_ALIAS_TO_ACCESS)

DEFINE QALIAS(USE_FOR_PUTS) GET(DISABLED)
       PUT(ENABLED) TARGQ(USE_ALIAS_TO_ACCESS)
```

You must also make the following ESM definitions:

```
TSS ADD(MQOWNER) MQQUEUE(ssid.USE_ALIAS_TO_ACCESS)
TSS ADD(MQOWNER) MQQUEUE(ssid.USE_FOR_GETS)
TSS ADD(MQOWNER) MQQUEUE(ssid.USE_FOR_PUTS)
```

Then, you must ensure that no users have access to the queue ssid.USE_ALIAS_TO_ACCESS, and give the appropriate users access to the alias. You can do this using the following ESM commands:

```
TSS PER(JOHNS) MQQUEUE(ssid.USE_FOR_GETS) ACC(READ, UPDATE)
TSS PER(JANED) MQQUEUE(ssid.USE_FOR_PUTS) ACC(READ, UPDATE)
```

This means that user JOHNS is only allowed to get messages from the USE_ALIAS_TO_ACCESS queue through the alias USE_FOR_GETS, and user JANED is only allowed to put messages through the alias queue USE_FOR_PUTS.

If you want to use a technique like this, you must inform the application developers, so that they can design their programs appropriately.

Security and remote queues

When a message is put on a remote queue, a security check is performed against the name of the remote queue. There is no check against the transmission queue identified by the remote queue definition.

This means that users accessing a remote queue need at least UPDATE authority to the resource, because it is not possible to browse a remote queue.

For example, you could define a remote queue as follows (this definition is based on OS/2 formats, and you should use the MQSeries for VSE/ESA Master Terminal transaction to create appropriate definitions):

```
DEFINE QREMOTE(BANK7.CREDIT.REFERENCE)
       RNAME(CREDIT.SCORING.REQUEST)
       RQMNAME(BNK7)
       XMITQ(BANK1.TO.BANK7)
```

For user JOHNS to put a message to the remote queue, you would need to grant the following access:

```
TSS PER(JOHNS) MQQUEUE(VSE.QM1.BANK7.CREDIT.REFERENCE) ACC(UPDATE)
```

where VSE.QM1 is the local MQSeries for VSE/ESA queue manager name.

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 be put 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.

The only application able to retrieve messages from the dead-letter queue should be a 'special' application that processes the undelivered messages. You can grant access to the dead-letter queue in the same way as any other queue.

The MQSeries for VSE/ESA Receiver and Sender MCAs user also requires UPDATE authority to the dead-letter queue. The MCAs run as the MQSeries for VSE/ESA startup user (that is, the user who starts MQSeries for VSE/ESA using MQSE, MQIT or MQMT). If a message cannot be delivered by either MCA, depending on the channel definition, the MCA may attempt to put the message to the dead-letter queue. Therefore, the MCA user must have UPDATE authority. For example:

```
TSS PER(MQSTART) MQQUEUE(VSE.QM1.DEAD.LETTER.QUEUE) ACC(UPDATE)
```

where VSE.QM1 is the local queue manager name.

If you want to use application programs that can put messages to, or get messages from, the dead-letter queue (or do both), you might consider using aliases, as described in "Using alias queues with MQGET and MQPUT" on page 293.

System queue security

System queues are accessed by the ancillary parts of the queue manager. System queues, in addition to the dead-letter queue, include:

- System log
- System monitor

Messages are put to the system log by the MQER transaction. This transaction runs as either the CICS default user identified by the CICS SIT parameter, or the user specified in the MQER DCT entry (see Chapter 2, "Installation," on page 7 for more details). Therefore, whichever of these users is configured to put messages to the system log should be granted connection, and also UPDATE authority, to the queue resource.

For example:

```
TSS PER(MQSYS) MQCONN(VSE.QM1.CICS) ACC(READ)
TSS PER(MQSYS) MQQUEUE(VSE.QM1.SYSTEM.LOG) ACC(UPDATE)
```

For performance reasons, messages written to the system monitor are handled internally to MQSeries for VSE/ESA. This means that no explicit authority is required for any particular user to put messages to the system monitor queue via normal MQSeries for VSE/ESA monitoring. If an application needs to explicitly put messages to the system monitor, the application user must have UPDATE authority to the queue resource.

If an application needs to get messages from the system monitor, the application user must have READ or UPDATE authority to the queue resource. For example:

```
TSS PER(JOHNS) MQQUEUE(ssid.SYSTEM.MONITOR) ACC(UPDATE)
```

Reply queue security

MQSeries for VSE/ESA supports messages with report types of Confirm-On-Arrival (COA) and Confirm-On-Delivery (COD). In either of these cases, a report message is generated by MQSeries for VSE/ESA. The required user authority varies, depending on whether the report message is for COA or COD, and how the ReplyToQ and ReplyToQMgr fields in the MQMD are used.

User authority for COA: When the ReplyToQ of the object message is a local queue to the VSE queue manager, the application user that puts the object message must have UPDATE authority to the ReplyToQ.

When the ReplyToQ of the object message is a remote queue name of the VSE queue manager, the application user that puts the object message must have UPDATE authority to the ReplyToQ.

When the ReplyToQ identifies a local queue on a remote queue manager, and the ReplyToQMgr identifies a remote queue manager, the application user that puts the object message must have UPDATE authority to the remote queue name that resolves the remote local queue and remote queue manager.

User authority for COD: When the ReplyToQ of the object message is a local queue to the VSE queue manager, the application user that gets the object message must have UPDATE authority to the ReplyToQ.

When the ReplyToQ of the object message is a remote queue name of the VSE queue manager, the application user that gets the object message must have UPDATE authority to the ReplyToQ.

When the ReplyToQ identifies a local queue on a remote queue manager, and the ReplyToQMgr identifies a remote queue manager, the application user that gets the object message must have UPDATE authority to the remote queue name that resolves the remote local queue and remote queue manager.

User authority for EXPIRY: Message expiry is managed by the queue manager when an application attempts to retrieve a message from a queue. At this time, the queue manager examines the Expiry field in the message descriptor of the message to determine whether or not the message has expired. If the message has expired, the queue manager continues to search for a valid message to return to the application. Expired messages are never returned.

When a message is identified as 'expired', the queue manager places an expiry entry on transient data queue MQXP. The MQXP data queue is defined at installation time to automatically fire a transaction (also MQXP) when there are items on the queue. For more information about installation and the MQXP transient data queue, refer to "Changing the MQXP TDQ definition" on page 14.

The MQXP transaction is responsible for logically deleting expired messages from queues. It is also responsible for generating expiry report messages when requested.

An expiry report message is requested when the Report field of the message descriptor of the originalmessage indicates one of the following report options:

MQRO_EXPIRATION

MQRO_EXPIRATION_WITH_DATA

MQRO_EXPIRATION_WITH_FULL_DATA

Expiry report messages are sent to the queue identified by the ReplyToQ and ReplyToQMgr message descriptor fields of expired messages. Consequently, the user that runs the MQXP transaction must have connect authority and the authority to put a report message on any potential reply queue.

The MQXP transaction runs as the CICS default user unless the destination control table (DCT) entry for the MQXP transient data queue identifies a specific userid in its definition (see "Changing the MQXP TDQ definition" on page 14 for more details).

Resource definitions for command security

If command security is active, you must define resources in the MQCMDS class, and permit the necessary groups or user IDs access to these resources, so that they can issue MQSeries commands.

Resource names for command security take the form:

```
ssid.command
```

where *command* is a type of command. For example:

```
ssid.ALTER.QLOCAL
ssid.DISPLAY.QMGR
ssid.DELETE.CHANNEL
```

Commands can be issued as:

- PCF messages
- MQSC verb-based commands
- Master terminal interactive options

Command security for PCF messages

The MQSeries command server (long-running transaction MQCS) starts an instance of the MQSeries command processor (transaction MQCX) for each PCF messages retrieved from the system command queue that passes initial validation.

If command security is active, the command server starts with MQCX transaction as the user identified in the UserIdentifier field of the message descriptor of the PCF message. Command security checks are then made for the user running the MQCX transaction. Consequently, command security checks are made against the user running the MQCX transaction, not the user running the MQCS transaction.

For command security to work in this way, the MQSeries command server (MQCS) must be started by a user with surrogate authority for all users that can put messages to the system command queue.

Authority for users that send command messages to the system command queue is three-fold:

- They must have authority to issue the command (for example, DISPLAY)
- They must have authority to send messages the ReplyToQ/ReplyToQMgr
- They must have authority to issue the command against a specific resource

This last requirement is only relevant for those commands that manipulate a specific resources, and falls under command resource security described below.

The authority required to issue PCF commands is described in the following table.

Table 15. Command authority for PCF commands

PCF Command	Resource	Authority
MQCMD_CHANGE_CHANNEL	ssid.ALTER.CHANNEL	ALTER
MQCMD_CHANGE_Q_MGR	ssid.ALTER.QMGR	ALTER
MQCMD_CHANGE_Q (alias)	ssid.ALTER.QALIAS	ALTER
MQCMD_CHANGE_Q (local)	ssid.ALTER.QLOCAL	ALTER
MQCMD_CHANGE_Q (remote)	ssid.ALTER.QREMOTE	ALTER
MQCMD_COPY_CHANNEL	ssid.DISPLAY.CHANNEL	READ
	ssid.DEFINE.CHANNEL	ALTER
MQCMD_COPY_Q (alias)	ssid.DISPLAY.QUEUE	READ
	ssid.DEFINE.QALIAS	ALTER
MQCMD_COPY_Q (local)	ssid.DISPLAY.QUEUE	READ
	ssid.DEFINE.QLOCAL	ALTER
MQCMD_COPY_Q (remote)	ssid.DISPLAY.QUEUE	READ
	ssid.DEFINE.QREMOTE	ALTER
MQCMD_CREATE_CHANNEL	ssid.DEFINE CHANNEL	ALTER
MQCMD_CREATE_Q (alias)	ssid.DEFINE.QALIAS	ALTER
MQCMD_CREATE_Q (local)	ssid.DEFINE.QLOCAL	ALTER

Table 15. Command authority for PCF commands (continued)

PCF Command	Resource	Authority
MQCMD_CREATE_Q (remote)	ssid.DEFINE.QALIAS	ALTER
MQCMD_DELETE_CHANNEL	ssid.DELETE.CHANNEL	ALTER
MQCMD_DELETE_Q (alias)	ssid.DELETE.QALIAS	ALTER
MQCMD_DELETE_Q (local)	ssid.DELETE.QLOCAL	ALTER
MQCMD_DELETE_Q (remote)	ssid.DELETE.QREMOTE	ALTER
MQCMD_INQUIRE_CHANNEL	ssid.DISPLAY.CHANNEL	READ
MQCMD_INQUIRE_Q	ssid.DISPLAY.QUEUE	READ
MQCMD_INQUIRE_Q_MGR	ssid.DISPLAY.QMGR	READ
MQCMD_INQUIRE_CHANNEL_NAMES	ssid.DISPLAY.CHANNEL	READ
MQCMD_INQUIRE_Q_NAMES	ssid.DISPLAY.QUEUE	READ
MQCMD_PING_Q_MGR	ssid.PING.QMGR	CONTROL
MQCMD_RESET_CHANNEL	ssid.RESET.CHANNEL	CONTROL
MQCMD_START_CHANNEL	ssid.START.CHANNEL	CONTROL
MQCMD_START_CHANNEL_LISTENER	ssid.START.LISTENER	CONTROL
MQCMD_STOP_CHANNEL	ssid.STOP.CHANNEL	CONTROL

Command security for MQSeries commands

MQSeries commands are generated by the MQSeries MQSC command utility. The MQSC utility generates PCF Escape messages from SYSIPT batch input and places them on the system command queue using the MQSeries batch interface. Consequently, the user that runs the MQSC command utility must have authority to connect to the queue manager and put messages on the system command queue.

Once an MQSC command has been placed on the system command queue (as a PCF Escape message) it is treated like any other PCF message. The MQSeries command server retrieves the message and starts the MQSeries command processor (MQCX transaction) as the user identified in the UserIdentifier field of the message descriptor.

If security is active, the user that submits the batch job to run the MQSC command utility is the userid that is placed in the UserIdentifier field. This user must have authority to put a reply message on the system reply queue. The system reply queue name is configurable as a communication parameter of the global system definition.

The authority required for PCF Escape messages is dependent on the verb-based text of the MQSC command embedded in the Escape message, and is determined by the following table:

Table 16. Command authority for MQSeries commands

MQSeries Command	Resource	Authority
ALTER CHANNEL	ssid.ALTER.CHANNEL	ALTER
ALTER QMGR	ssid.ALTER.QMGR	ALTER
ALTER QALIAS	ssid.ALTER.QALIAS	ALTER
ALTER QLOCAL	ssid.ALTER.QLOCAL	ALTER

Table 16. Command authority for MQSeries commands (continued)

MQSeries Command	Resource	Authority
ALTER QREMOTE	ssid.ALTER.QREMOTE	ALTER
DEFINE CHANNEL	ssid.DEFINE CHANNEL	ALTER
DEFINE QALIAS	ssid.DEFINE.QALIAS	ALTER
DEFINE QLOCAL	ssid.DEFINE.QLOCAL	ALTER
DEFINE QREMOTE	ssid.DEFINE.QALIAS	ALTER
DELETE CHANNEL	ssid.DELETE.CHANNEL	ALTER
DELETE QALIAS	ssid.DELETE.QALIAS	ALTER
DELETE QLOCAL	ssid.DELETE.QLOCAL	ALTER
DELETE QREMOTE	ssid.DELETE.QREMOTE	ALTER
DISPLAY CHANNEL	ssid.DISPLAY.CHANNEL	READ
DISPLAY QALIAS	ssid.DISPLAY.QUEUE	READ
DISPLAY QLOCAL	ssid.DISPLAY.QUEUE	READ
DISPLAY QREMOTE	ssid.DISPLAY.QUEUE	READ
DISPLAY QMGR	ssid.DISPLAY.QMGR	READ
PING QMGR	ssid.PING.QMGR	CONTROL
RESET CHANNEL	ssid.RESET.CHANNEL	CONTROL
START CHANNEL	ssid.START.CHANNEL	CONTROL
START LISTENER	ssid.START.LISTENER	CONTROL
STOP CHANNEL	ssid.STOP.CHANNEL	CONTROL

Command security for MQMT options

Command can be issued interactively via the MQSeries master terminal transaction. Generally, these are invoked in native CICS from the MQMT transaction.

The MQMT transaction provides a primary options menu. Most of the menu options can be invoked directly by starting the appropriate MQSeries transaction.

The following table describes these options and transactions, and the resources and authority necessary to perform the option.

Table 17. Command authority for MQMT options

Option	Trans	Function	Resource	Authority
1.1	MQMS	Maintain QMgr	ssid.DISPLAY.QMGR	READ
			ssid.ALTER.QMGR	ALTER
1.2	MQMQ	Maintain Queues	ssid.DISPLAY.QUEUE	READ
1.2	MQMQ	Maintain Queue (alias)	ssid.ALTER.QALIAS	ALTER
1.2	MQMQ	Maintain Queue (local)	ssid.ALTER.QLOCAL	ALTER
1.2	MQMQ	Maintain Queue (remote)	ssid.ALTER.QREMOTE	ALTER
1.3	MQMH	Maintain Channel	ssid.DISPLAY.CHANNEL	READ
			ssid.ALTER.CHANNEL	ALTER
1.5	MQDS	Display QMgr	ssid.DISPLAY.QMGR	READ

Table 17. Command authority for MQMT options (continued)

Option	Trans	Function	Resource	Authority
1.6	MQDQ	Display Queues	ssid.DISPLAY.QUEUE	READ
1.7	MQDH	Display Channel	ssid.DISPLAY.CHANNEL	READ
2.2	MQMB	Start channel	ssid.START.CHANNEL	CONTROL
2.2	MQMB	Stop channel	ssid.STOP.CHANNEL	CONTROL
2.3	MQMR	Reset channel	ssid.RESET.CHANNEL	CONTROL
3.1	MQQM	Monitor queues	ssid.DISPLAY.QUEUE	READ
3.2	MQCM	Monitor channel	ssid.DISPLAY.CHANNEL	READ

In addition to the command authority required to issue a command, the issuing user must have command resource authority for a specific resource when a specific resource is affected by the command.

Resource definitions for command resource security

If command resource security is active, you must grant command authority by resource to any user that is authorized to issue commands against that specific resource.

Command security allows a user to issue certain commands. Command resource security allows a user to issue those commands against specific resources. For example, a user may be authorized to 'DISPLAY' certain queues and not others. To achieve this, the user is granted command authority to 'DISPLAY', and is then granted command resource security for each queue the user is allowed to display.

Resources relevant to command resource security must be defined in the MQADMIN class.

The following table describes the resources and authority required for PCF messages (the constant 'ssid' should be replaced by the queue manager name of the local queue manager, and names expressed in lower-case should be replaced with the names of specific resources):

Table 18. Command resource authority for PCF commands

Command	Command resource	Authority
MQCMD_CHANGE_CHANNEL	ssid.CHANNEL.channel	ALTER
MQCMD_CHANGE_Q	ssid.QUEUE.queue	ALTER
MQCMD_COPY_CHANNEL	ssid.CHANNEL.tochannel	ALTER
MQCMD_COPY_Q	ssid.QUEUE.toqueue	ALTER
MQCMD_CREATE_CHANNEL	ssid.CHANNEL.channel	ALTER
MQCMD_CREATE_Q	ssid.QUEUE.queue	ALTER
MQCMD_DELETE_CHANNEL	ssid.CHANNEL.channel	ALTER
MQCMD_DELETE_Q	ssid.QUEUE.queue	ALTER
MQCMD_RESET_CHANNEL	ssid.CHANNEL.channel	CONTROL
MQCMD_START_CHANNEL	ssid.CHANNEL.channel	CONTROL
MQCMD_STOP_CHANNEL	ssid.CHANNEL.channel	CONTROL

The following table describes the resources and authority required for MQCS commands (the constant 'ssid' should be replaced by the queue manager name of the local queue manager, and names expressed in lower-case should be replaced with the names of specific resources):

Table 19. Command resource authority for MQSeries commands

MQCS Command	Resource	Authority
ALTER CHANNEL	ssid.CHANNEL.channel	ALTER
ALTER QALIAS	ssid.QUEUE.queue	ALTER
ALTER QLOCAL	ssid.QUEUE.queue	ALTER
ALTER QREMOTE	ssid.QUEUE.queue	ALTER
DEFINE CHANNEL	ssid.CHANNEL.channel	ALTER
DEFINE QALIAS	ssid.QUEUE.queue	ALTER
DEFINE QLOCAL	ssid.QUEUE.queue	ALTER
DEFINE QREMOTE	ssid.QUEUE.queue	ALTER
DELETE CHANNEL	ssid.CHANNEL.channel	ALTER
DELETE QALIAS	ssid.QUEUE.queue	ALTER
DELETE QLOCAL	ssid.QUEUE.queue	ALTER
DELETE QREMOTE	ssid.QUEUE.queue	ALTER
RESET CHANNEL	ssid.CHANNEL.channel	CONTROL
START CHANNEL	ssid.CHANNEL.channel	CONTROL
STOP CHANNEL	ssid.CHANNEL.channel	CONTROL

The following table describes the resources and authority required for master terminal options and associated transactions (the constant 'ssid' should be replaced by the queue manager name of the local queue manager, and names expressed in lower-case should be replaced with the names of specific resources):

Table 20. Command resource authority for MQMT options

Option	Trans	Function	Resource	Authority
1.2	MQMQ	Maintain Queues	ssid.QUEUE.queue	ALTER
1.3	MQMH	Maintain Channel	ssid.CHANNEL.channel	ALTER
2.2	MQMB	Start channel	ssid.CHANNEL.channel	CONTROL
2.2	MQMB	Stop channel	ssid.CHANNEL.channel	CONTROL
2.3	MQMR	Reset channel	ssid.CHANNEL.channel	CONTROL

In addition to the command resource authority described in table x, the master terminal transactions also require special authority for options 2.5 and 4.0. Relevant resources must be defined in the MQQUEUE class (not the MQADMIN class) according to the following table:

Table 21. Command resource authority for MQMT options 2.5 and 4.0

Option	Trans	Function	Resource	Authority
2.5	MQMD	Maintain messages	ssid.queue	UPDATE
4.0	MQBQ	Browse queues	ssid.queue	READ

Security implementation checklist

This section contains 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 "Using security classes and resources" on page 288.

If you require security checking to be implemented on at least one of your MQSeries subsystems, you must first activate the MQADMIN class. Then, for each MQSeries subsystem, you must decide whether you need security checking on that subsystem. If you do not require security checking, you must define an ssid.NO.SUBSYS.SECURITY profile in the MQADMIN class.

If you do require security checking, use the following checklist to implement it:

- Do you need connection security?

  Yes: Define appropriate connection profiles in the MQCONN class and permit the appropriate users or groups access to these profiles.

  No: Define an ssid.NO.CONNECT.CHECKS resource in the MQADMIN class and grant your MQSeries for VSE/ESA startup user READ access to the resource.

- Do you need security checking on commands?

  Yes: Activate the MQCMDS class. Define appropriate command profiles in the MQCMDS class and permit the appropriate users access to these profiles. If command authority by individual resource is required, define appropriate resource profiles in the MQADMIN class and grant access to these profiles as appropriate.

  No: Define resources ssid.NO.CMDS.CHECKS and ssid.NO.CMD.RESC.CHECKS in the MQADMIN class and grant read authority to both resources to your MQSeries for VSE/ESA startup user.

- Do you need queue security?

  Yes: Activate the MQQUEUE class. Define appropriate queue resources in the MQQUEUE class and permit the appropriate user access to these profiles.

  Also, ensure that your MQSeries VSAM datasets are protected against unauthorized access.

  No: Define an ssid.NO.QUEUE.CHECKS profile in the MQADMIN class and grant read authority to your MQSeries for VSE/ESA startup user.

- Do you plan to have remote client connections?

  Yes: Ensure that your remote clients use the MQ_USER_ID and MQ_PASSWORD environment variables, and ensure that such users are defined to your ESM.

For a detailed example of resource definitions and access authorities for MQSeries for VSE/ESA, refer to Appendix H, "Security implementation," on page 499.

Appendix A. CICS control table definitions

This appendix contains various sample entries for the CICS control tables.

Sample file control table entries

FCT macro definitions are only necessary for MQSeries for VSE/ESA running under CICS/VSE. Matching file definitions under CICS TS are part of the CICS CSD, and are defined using the DEFINE FILE CSD command. Consequently, the following sample is for MQ under CICS/VSE only.

```
*--------------------------------------------------------------------*
* Licensed Materials - Property of IBM                               *
* 5686-A06                                                           *
* (C) Copyright IBM Corp. 1998, 2002                                 *
*                                                                    *
* US Government Users Restricted Rights - Use, duplication or        *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.  *
*--------------------------------------------------------------------*
*
*--------------------------------------------------------------------*
*         Start of  MQSeries VSAM cluster definitions                *
*                                                                    *
*  For performance reasons entries may be modified to add LSRPOOL    *
*  explicit specifications.                                          *
*--------------------------------------------------------------------*
*
* system setup file
MQFSSET  DFHFCT TYPE=DATASET,DATASET=MQFSSET,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,BROWSE),                                 *
               LOG=NO,                                                *
               RSL=PUBLIC,                                            *
               STRNO=5,                                               *
               RECFORM=(FIXED,BLOCKED)
* configuration file
MQFCNFG  DFHFCT TYPE=DATASET,DATASET=MQFCNFG,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               LOG=YES,                                               *
               RSL=PUBLIC,                                            *
               STRNO=20,                                              *
               RECFORM=(FIXED,BLOCKED)
*--reorganization file
MQFREOR  DFHFCT TYPE=DATASET,DATASET=MQFREOR,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               RSL=PUBLIC,                                            *
               LOG=NO,                                                *
               STRNO=16,                                              *
               RECFORM=(VARIABLE,BLOCKED)
*--example of queues (input followed by output)
MQFI001  DFHFCT TYPE=DATASET,DATASET=MQFI001,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
               RSL=PUBLIC,                                            *
               LOG=YES,                                               *
               STRNO=16,                                              *
               RECFORM=(VARIABLE,BLOCKED)
MQFO001  DFHFCT TYPE=DATASET,DATASET=MQFO001,                          *
               ACCMETH=VSAM,                                          *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),               *
```

© Copyright IBM Corp. 1995, 2004

```
               LOG=YES,                                                 *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
MQFI002  DFHFCT TYPE=DATASET,DATASET=MQFI002,                           *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               RSL=PUBLIC,                                              *
               LOG=YES,                                                 *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
MQFO002  DFHFCT TYPE=DATASET,DATASET=MQFO002,                           *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               LOG=YES,                                                 *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
MQFI003  DFHFCT TYPE=DATASET,DATASET=MQFI003,                           *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               LOG=YES,                                                 *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
MQFO003  DFHFCT TYPE=DATASET,DATASET=MQFO003,                           *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               LOG=YES,                                                 *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
*--SYSTEM DEFINITIONS
MQFLOG   DFHFCT TYPE=DATASET,DATASET=MQFLOG,                            *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               LOG=YES,                                                 *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
MQFERR   DFHFCT TYPE=DATASET,DATASET=MQFERR,                            *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               LOG=YES,                                                 *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
MQFMON   DFHFCT TYPE=DATASET,DATASET=MQFMON,                            *
               ACCMETH=VSAM,                                            *
               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),                 *
               LOG=NO,                                                  *
               RSL=PUBLIC,                                              *
               STRNO=16,                                                *
               RECFORM=(VARIABLE,BLOCKED)
*----------------------------------------------------------------------*
*             End  of MQSeries VSAM cluster definitions              *
*----------------------------------------------------------------------*
```

The following sample definitions are also provided for the Programmable Command Formats (PCF) and MQSeries Command (MQSC) features:

```
*--PCF SYSTEM DEFINITIONS
* MQFACMD  DFHFCT TYPE=DATASET,DATASET=MQFACMD,
*               ACCMETH=VSAM,
*               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),
*               LOG=YES,
*               RSL=PUBLIC,
```

```
*               STRNO=16,
*               RECFORM=(VARIABLE,BLOCKED)
* MQFARPY  DFHFCT TYPE=DATASET,DATASET=MQFARPY,
*               ACCMETH=VSAM,
*               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),
*               LOG=YES,
*               RSL=PUBLIC,
*               STRNO=16,
*               RECFORM=(VARIABLE,BLOCKED)
```

Similarly, the following sample definitions are also provided for the Instrumentation Events feature:

```
*--INSTRUMENTATION EVENT DEFINITIONS
* MQFIEQE  DFHFCT TYPE=DATASET,DATASET=MQFIEQE,
*               ACCMETH=VSAM,
*               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),
*               LOG=YES,
*               RSL=PUBLIC,
*               STRNO=16,
*               RECFORM=(VARIABLE,BLOCKED)
* MQFIECE  DFHFCT TYPE=DATASET,DATASET=MQFIECE,
*               ACCMETH=VSAM,
*               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),
*               LOG=YES,
*               RSL=PUBLIC,
*               STRNO=16,
*               RECFORM=(VARIABLE,BLOCKED)
* MQFIEPE  DFHFCT TYPE=DATASET,DATASET=MQFIEPE,
*               ACCMETH=VSAM,
*               SERVREQ=(READ,UPDATE,ADD,BROWSE,DELETE),
*               LOG=YES,
*               RSL=PUBLIC,
*               STRNO=16,
*               RECFORM=(VARIABLE,BLOCKED)
```

Sample destination control table entry

Destination control table entry MQER is required in order for MQSeries system messages to be logged to the SYSTEM.LOG queue.

In the event that security is installed for the MQSeries subsystem, the DCT entry may need to be modified to include a "USERID=user" parameter. For information on this requirement, refer to "Changing the MQER TDQ definition" on page 13.

Similarly, destination control table entry MQXP is required for the queue manager to manage expired messages.

Once again, if security is installed, the DCT entry may need to be modified according to the instructions described in section "Changing the MQXP TDQ definition" on page 14.

```
*--------------------------------------------------------------------*
* Licensed Materials - Property of IBM                               *
*                                                                    *
* 5686-A06                                                           *
* (C) Copyright IBM Corp. 1998, 2002.                                *
*                                                                    *
* US Government Users Restricted Rights - Use, duplication or        *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.  *
*--------------------------------------------------------------------*
*
*--------------------------------------------------------------------*
*   START  OF MQSERIES  DCT ENTRIES
```

Sample DCT

```
*-------------------------------------------------------------------*
*
*        MQER - System Log transient data queue
*
MQER     DFHDCT TYPE=INTRA,                                         *
               RSL=PUBLIC,                                          *
               DESTID=MQER,                                         *
               DESTFAC=FILE,                                        *
               TRANSID=MQER,                                        *
               TRIGLEV=1
*
*        MQXP - Message expiry transient data queue
*
MQXP     DFHDCT TYPE=INTRA,                                         *
               RSL=PUBLIC,                                          *
               DESTID=MQXP,                                         *
               DESTFAC=FILE,                                        *
               TRANSID=MQXP,                                        *
               TRIGLEV=1
*
*        MQIE - Instrumentation Events transient data queue
*
MQIE     DFHDCT TYPE=INTRA,                                         *
                RSL=PUBLIC,                                         *
                DESTID=MQIE,                                        *
                DESTFAC=FILE,                                       *
                TRANSID=MQIE,                                       *
                TRIGLEV=1
*-------------------------------------------------------------------*
*   END  OF MQSERIES DCT ENTRIES
*-------------------------------------------------------------------*
```

Sample JCL file definition for CICS deck

```
*-------------------------------------------------------------------*
* Licensed Materials - Property of IBM                              *
* 5686-A06                                                          *
* (C) Copyright IBM Corp. 1998, 2002                                 *
*                                                                   *
* US Government Users Restricted Rights - Use, duplication or       *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. *
*-------------------------------------------------------------------*
*   Sample JCL file definition for CICS deck                        *
*   The DLBL statements in this JCL correspond to entries in CICSFCT*
*   therefore if there are any new file ids to be added in here,    *
*   it must also be added into the corresponding JCL                *
*                                                                   *
*   Fields marked with ?volid? and ?cat-name? must be changed to    *
*   suit customer own site specifications.                          *
*-------------------------------------------------------------------*
// DLBL MQFSSET,'MQSERIES.MQFSSET',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFCNFG,'MQSERIES.MQFCNFG',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFREOR,'MQSERIES.MQFREOR',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFI001,'MQSERIES.MQFI001',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFI002,'MQSERIES.MQFI002',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFI003,'MQSERIES.MQFI003',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFO001,'MQSERIES.MQFO001',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFO002,'MQSERIES.MQFO002',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFO003,'MQSERIES.MQFO003',0,VSAM,CAT=?cat-name?
```

```
// EXTENT ,?volid?
// DLBL MQFERR,'MQSERIES.MQFERR',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFLOG,'MQSERIES.MQFLOG',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
// DLBL MQFMON,'MQSERIES.MQFMON',0,VSAM,CAT=?cat-name?
// EXTENT ,?volid?
/. c if you are using PCF then also customize following 2 labels
/. DLBL MQFACMD,'MQSERIES.MQFACMD',0,VSAM,CAT=?cat-name?
/. EXTENT ,?volid?
/. DLBL MQFARPY,'MQSERIES.MQFARPY',0,VSAM,CAT=?cat-name?
/. EXTENT ,?volid?
/. c if you require Instrumentation Events, customize following labels
/. DLBL MQFIEQE,'MQSERIES.MQFIEQE',0,VSAM,CAT=?cat-name?
/. EXTENT ,?volid?
/. DLBL MQFIECE,'MQSERIES.MQFIECE',0,VSAM,CAT=?cat-name?
/. EXTENT ,?volid?
/. DLBL MQFIEPE,'MQSERIES.MQFIEPE',0,VSAM,CAT=?cat-name?
/. EXTENT ,?volid?
*--------------------------------------------------------------------*
*   End of sample jcl file definition for cics deck                  *
*--------------------------------------------------------------------*
```

Sample JCL to create CICS CSD group

```
* ** JOB JNM=MQJCSD,CLASS=0,DISP=D
* ** LST DISP=H,CLASS=Q,PRI=3
// JOB MQJCSD Define resources for MQ/Series for VSE/ESA to CICS CSD.
* -------------------------------------------------------------------*
*   Please change :                                                  *
*                 "*  ** JOB" to  "* $$ JOB"                         *
*                 "*  ** LST" to  "* $$ LST"                         *
*                 "*  ** EOJ" to  "* $$ EOJ"                         *
* -------------------------------------------------------------------*
*       Create CICS CSD group for MQ/Series VSE/ESA                  *
*                                                                    *
*   This file is a sample and may need modifications to suit the     *
*   users environment (eg. Group name, or list name).                *
* -------------------------------------------------------------------*
* Licensed Materials - Property of IBM                               *
*                                                                    *
* 5686-A06                                                           *
* (C) Copyright IBM Corp. 1998, 2002                                 *
*                                                                    *
* US Government Users Restricted Rights - Use, duplication or        *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.  *
* -------------------------------------------------------------------*
// EXEC DFHCSDUP
*--------------------------------------------------------------------*
*            Definitions for MQ/Series VSE/ESA                       *
*--------------------------------------------------------------------*
*
DELETE GROUP(MQM)
*
*--                      Definitions of MQ/Series Programs
*
 DEFINE  PROGRAM(MQPSTBI ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSPBI ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQBICIRH) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQBICITK) GROUP(MQM) LANGUAGE(ASSEMBLER) RSL(PUBLIC)  *
         RESIDENT(YES)
 DEFINE  PROGRAM(MQPMTP  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMCFG ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMMON ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMOPR ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPDISP ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMSYS ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
```

```
 DEFINE  PROGRAM(MQPMQUE ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMCHN ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMSS  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMSC  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMMSN ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMSI  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMDEL ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMMOQ ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMMOC ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPMCPG ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPXQC  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
*-- NON-ADMINISTRATOR
 DEFINE  PROGRAM(MQPAIP0 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPAIP1 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPAIP2 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSEND ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPRECV ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPRPRT ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPCCKPT) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPQUE1 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPQUE2 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPECHO ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPINIT1) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPINIT2) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSSQ  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSCHK ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPERR  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPFINDQ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPQDEL ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSTOP ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSTART) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSREC ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPQREC ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSMAP ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSSET ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPSENV ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPCMD  ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPVSAM ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPCNSL ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPEXPR ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPIEVT ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(MQPTCPLN) GROUP(MQM) LANGUAGE(C)         RSL(PUBLIC)
 DEFINE  PROGRAM(MQPTCPSV) GROUP(MQM) LANGUAGE(C)         RSL(PUBLIC)
 DEFINE  PROGRAM(MQPCABND) GROUP(MQM) LANGUAGE(C)         RSL(PUBLIC)
 DEFINE  PROGRAM(MQPCSRV)  GROUP(MQM) LANGUAGE(C)         RSL(PUBLIC)
 DEFINE  PROGRAM(MQPCPRO)  GROUP(MQM) LANGUAGE(C)         RSL(PUBLIC)
 DEFINE  PROGRAM(MQPIDCMS) GROUP(MQM) LANGUAGE(ASSEMBLER)
 DEFINE  PROGRAM(DCHFMT4)  GROUP(MQM) LANGUAGE(C)
*-- MAPS
 DEFINE  MAPSET(MQMMTP  ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMCFG ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMMON ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMOPR ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMDISP ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMSYS ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMQUE ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMCHN ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMSS  ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMSC  ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMMSN ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMSI  ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMDEL ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMMOQ ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMMOC ) GROUP(MQM) RSL(PUBLIC)
 DEFINE  MAPSET(MQMMCPG ) GROUP(MQM) RSL(PUBLIC)
*-- TEST PROGRAMS
 DEFINE  PROGRAM(TTPTST1 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
```

```
 DEFINE  PROGRAM(TTPTST2 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  PROGRAM(TTPTST3 ) GROUP(MQM) LANGUAGE(COBOL)     RSL(PUBLIC)
 DEFINE  MAPSET(TTMTST3 )  GROUP(MQM) RSL(PUBLIC)
*
*--                     Definitions of MQ/Series Transactions
*
 DEFINE TRANSACTION(MQBI)  GROUP(MQM) PROGRAM(MQBICITK)
 DEFINE TRANSACTION(MQBX)  GROUP(MQM) PROGRAM(MQBICIRH)
 DEFINE TRANSACTION(MQMT)  GROUP(MQM) PROGRAM(MQPMTP)
 DEFINE TRANSACTION(MQMC)  GROUP(MQM) PROGRAM(MQPMCFG)
 DEFINE TRANSACTION(MQMO)  GROUP(MQM) PROGRAM(MQPMOPR)
 DEFINE TRANSACTION(MQMM)  GROUP(MQM) PROGRAM(MQPMMON)
 DEFINE TRANSACTION(MQBQ)  GROUP(MQM) PROGRAM(MQPDISP)
 DEFINE TRANSACTION(MQMS)  GROUP(MQM) PROGRAM(MQPMSYS)
 DEFINE TRANSACTION(MQDS)  GROUP(MQM) PROGRAM(MQPMSYS)
 DEFINE TRANSACTION(MQMQ)  GROUP(MQM) PROGRAM(MQPMQUE)
 DEFINE TRANSACTION(MQDQ)  GROUP(MQM) PROGRAM(MQPMQUE)
 DEFINE TRANSACTION(MQMH)  GROUP(MQM) PROGRAM(MQPMCHN)
 DEFINE TRANSACTION(MQDH)  GROUP(MQM) PROGRAM(MQPMCHN)
 DEFINE TRANSACTION(MQMA)  GROUP(MQM) PROGRAM(MQPMSS)
 DEFINE TRANSACTION(MQMB)  GROUP(MQM) PROGRAM(MQPMSC)
 DEFINE TRANSACTION(MQMR)  GROUP(MQM) PROGRAM(MQPMMSN)
 DEFINE TRANSACTION(MQMI)  GROUP(MQM) PROGRAM(MQPMSI)
 DEFINE TRANSACTION(MQMD)  GROUP(MQM) PROGRAM(MQPMDEL)
 DEFINE TRANSACTION(MQQM)  GROUP(MQM) PROGRAM(MQPMMOQ)
 DEFINE TRANSACTION(MQCM)  GROUP(MQM) PROGRAM(MQPMMOC)
 DEFINE TRANSACTION(MQIT)  GROUP(MQM) PROGRAM(MQPINIT1)
 DEFINE TRANSACTION(MQ02)  GROUP(MQM) PROGRAM(MQPAIP2)
 DEFINE TRANSACTION(MQ01)  GROUP(MQM) PROGRAM(MQPRECV)
 DEFINE TRANSACTION(MQ03)  GROUP(MQM) PROGRAM(MQPSEND)
 DEFINE TRANSACTION(MQSS)  GROUP(MQM) PROGRAM(MQPSSQ)
 DEFINE TRANSACTION(MQSM)  GROUP(MQM) PROGRAM(MQPSCHK)
 DEFINE TRANSACTION(MQER)  GROUP(MQM) PROGRAM(MQPERR)
 DEFINE TRANSACTION(MQQD)  GROUP(MQM) PROGRAM(MQPQDEL)
 DEFINE TRANSACTION(MQQA)  GROUP(MQM) PROGRAM(MQPQDEL)
 DEFINE TRANSACTION(MQST)  GROUP(MQM) PROGRAM(MQPSTOP)
 DEFINE TRANSACTION(MQSU)  GROUP(MQM) PROGRAM(MQPSSET)
 DEFINE TRANSACTION(MQSE)  GROUP(MQM) PROGRAM(MQPSENV)
 DEFINE TRANSACTION(MQSR)  GROUP(MQM) PROGRAM(MQPSREC)
 DEFINE TRANSACTION(MQSQ)  GROUP(MQM) PROGRAM(MQPQREC)
 DEFINE TRANSACTION(MQTL)  GROUP(MQM) PROGRAM(MQPTCPLN)
 DEFINE TRANSACTION(MQRG)  GROUP(MQM) PROGRAM(MQPVSAM)
 DEFINE TRANSACTION(MQMP)  GROUP(MQM) PROGRAM(MQPMCPG)
 DEFINE TRANSACTION(MQDP)  GROUP(MQM) PROGRAM(MQPMCPG)
 DEFINE TRANSACTION(MQCS)  GROUP(MQM) PROGRAM(MQPCSRV)
 DEFINE TRANSACTION(MQCX)  GROUP(MQM) PROGRAM(MQPCPRO)
 DEFINE TRANSACTION(MQCN)  GROUP(MQM) PROGRAM(MQPCNSL)
 DEFINE TRANSACTION(MQXP)  GROUP(MQM) PROGRAM(MQPEXPR)
 DEFINE TRANSACTION(MQIE)  GROUP(MQM) PROGRAM(MQPIEVT)
 DEFINE TRANSACTION(MQCL)  GROUP(MQM) PROGRAM(MQPCMD)
*-- Test Transactions
 DEFINE TRANSACTION(TST1)  GROUP(MQM) PROGRAM(TTPTST1)
 DEFINE TRANSACTION(TST2)  GROUP(MQM) PROGRAM(TTPTST2)
 DEFINE TRANSACTION(TST3)  GROUP(MQM) PROGRAM(TTPTST3)
*-- Add MQ/Series group to the standard VSE/ESA list.
 ADD GROUP(MQM) LIST(VSELIST)
/*
/&
* ** EOJ
```

For MQSeries for VSE/ESA running under CICS TS, File Control Table (FCT) definitions are now defined in the CSD. Consequently, the following additional definitions are required for a CICS TS environment:

```
*
*--                     CICS/TS Definitions for MQSeries Sample Files
*
```

```
DEFINE FILE(MQFSSET)    GROUP(MQM)
       DSNAME(MQSERIES.MQFSSET)
       TABLE(NO)        RECORDFORMAT(F)
       READ(YES)        BROWSE(YES)
       CATNAME(MQMCAT)  STRINGS(5)
       DATABUFFERS(6)
       LSRPOOLID(4)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFCNFG)    GROUP(MQM)
       DSNAME(MQSERIES.MQFCNFG)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(20)
       DATABUFFERS(25)  INDEXBUFFERS(20)
       LSRPOOLID(7)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFREOR)    GROUP(MQM)
       DSNAME(MQSERIES.MQFREOR)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(1)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(NONE)  RECOVERY(NONE)

DEFINE FILE(MQFI001)    GROUP(MQM)
       DSNAME(MQSERIES.MQFI001)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(1)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFI002)    GROUP(MQM)
       DSNAME(MQSERIES.MQFI002)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(2)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFI003)    GROUP(MQM)
       DSNAME(MQSERIES.MQFI003)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(3)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFO001)    GROUP(MQM)
       DSNAME(MQSERIES.MQFO001)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(1)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFO002)    GROUP(MQM)
       DSNAME(MQSERIES.MQFO002)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
```

```
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(2)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFO003)    GROUP(MQM)
       DSNAME(MQSERIES.MQFO003)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(3)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFLOG)     GROUP(MQM)
       DSNAME(MQSERIES.MQFLOG)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(5)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFMON)     GROUP(MQM)
       DSNAME(MQSERIES.MQFMON)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(NONE)  RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFERR)     GROUP(MQM)
       DSNAME(MQSERIES.MQFERR)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(NONE)  RECOVERY(BACKOUTONLY)
```

The following optional definitions are also provided for the Programmable Command Formats (PCF) and MQSeries Command (MQSC) features:

```
DEFINE FILE(MQFACMD)    GROUP(MQM)
       DSNAME(MQSERIES.MQFACMD)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(5)     RECOVERY(BACKOUTONLY)

DEFINE FILE(MQFARPY)    GROUP(MQM)
       DSNAME(MQSERIES.MQFARPY)
       TABLE(NO)        RECORDFORMAT(V)
       ADD(YES)         BROWSE(YES)    DELETE(YES)
       READ(YES)        UPDATE(YES)
       CATNAME(MQMCAT)  STRINGS(16)
       DATABUFFERS(20)  INDEXBUFFERS(16)
       LSRPOOLID(5)     RECOVERY(BACKOUTONLY)
```

Similarly, the following optional definitions are also provided for the Instrumentation Events feature:

```
DEFINE FILE(MQFIEQE)    GROUP(MQM)
       DSNAME(MQSERIES.MQFIEQE)
       TABLE(NO)        RECORDFORMAT(V)
```

```
|                              ADD(YES)         BROWSE(YES)    DELETE(YES)
|                              READ(YES)        UPDATE(YES)
|                              CATNAME(MQMCAT)  STRINGS(16)
|                              DATABUFFERS(20)  INDEXBUFFERS(16)
|                              LSRPOOLID(5)     RECOVERY(BACKOUTONLY)
|
|                       DEFINE FILE(MQFIECE)    GROUP(MQM)
|                              DSNAME(MQSERIES.MQFIECE)
|                              TABLE(NO)        RECORDFORMAT(V)
|                              ADD(YES)         BROWSE(YES)    DELETE(YES)
|                              READ(YES)        UPDATE(YES)
|                              CATNAME(MQMCAT)  STRINGS(16)
|                              DATABUFFERS(20)  INDEXBUFFERS(16)
|                              LSRPOOLID(5)     RECOVERY(BACKOUTONLY)
|
|                       DEFINE FILE(MQFIEPE)    GROUP(MQM)
|                              DSNAME(MQSERIES.MQFIEPE)
|                              TABLE(NO)        RECORDFORMAT(V)
|                              ADD(YES)         BROWSE(YES)    DELETE(YES)
|                              READ(YES)        UPDATE(YES)
|                              CATNAME(MQMCAT)  STRINGS(16)
|                              DATABUFFERS(20)  INDEXBUFFERS(16)
|                              LSRPOOLID(5)     RECOVERY(BACKOUTONLY)
```

Appendix B. Application Programming Reference

This appendix should be used in conjunction with the *MQSeries Application Programming Reference* manual.

Structure data types

The following structure data types are supported by MQSeries for VSE/ESA V2.1:

MQDLH	Dead letter header
MQGMO	Get message options
MQMD	Message descriptor
MQOD	Object descriptor
MQPMO	Put message options
MQTM	Trigger message 2

The declarations of these structures are as described in the *MQSeries Application Programming Reference*, with the following exceptions.

MQDLH – Dead-letter header

Fields

Version (MQLONG)
Structure version number.

The value must be:

MQDLH_VERSION_1
Version number for dead-letter header structure.

Reason (MQLONG)
Reason message arrived on dead-letter (undelivered-message) queue.

This identifies the reason why the message was placed on the dead-letter queue instead of on the original destination queue. It should be one of the MQRC_* values (for example, MQRC_Q_FULL).

The initial value of this field is MQRC_NONE.

PutApplType (MQLONG)
Type of application that put message on dead-letter (undelivered-message) queue.

This field has the same meaning as the *PutApplType* field in the message descriptor MQMD.

The value used by MQSeries for VSE/ESA is MQAT_CICS_VSE.

PutApplName (MQCHAR28)
Name of application that put message on dead-letter (undelivered-message) queue.

The format of the name is an eight character applid, followed by a four character tranid.

MQGMO – Get message options

The fields in the Version 1 structure only are supported.

© Copyright IBM Corp. 1995, 2004

Fields

Version (MQLONG)
Structure version number.

The value must be:

MQGMO_VERSION_1
Version-1 get-message options structure.

Options (MQLONG)
Options that control the action of MQGET.

The following options are supported:
MQGMO_WAIT
MQGMO_NO_WAIT
MQGMO_SYNCPOINT
MQGMO_BROWSE_FIRST
MQGMO_BROWSE_NEXT
MQGMO_ACCEPT_TRUNCATED_MSG
MQGMO_MSG_UNDER_CURSOR
MQGMO_LOCK
MQGMO_UNLOCK
MQGMO_CONVERT

Signal1 (MQLONG)
Signal.

This is a reserved field; its value is not significant.

Signal2 (MQLONG)
Signal identifier.

This is a reserved field; its value is not significant.

MQMD – Message descriptor

The fields in the Version 1 structure only are supported.

Fields

Version (MQLONG)
Structure version number.

The value must be:

MQMD_VERSION_1
Version-1 message descriptor structure.

Report (MQLONG)
Options for report messages.

This field identifies the required report options associated with the message.

The value must be one of the following:

MQRO_NONE
No report options required.

MQRO_COA
Confirm on arrival.

MQRO_COA_WITH_DATA
Confirm on arrival with 100 bytes of data.

MQRO_COA_WITH_FULL_DATA
Confirm on arrival with full message data.

MQRO_COD
Confirm on delivery.

MQRO_COD_WITH_DATA
Confirm on delivery with 100 bytes of data.

MQRO_COD_WITH_FULL_DATA
Confirm on delivery with full message data.

MQRO_COPY_MSG_ID_TO_CORREL_ID
Use MSG ID of message for CORREL ID of message.

MQRO_EXPIRATION
Report message expiration.

MQRO_EXPIRATION_WITH_DATA
Report message expiration with 100 bytes of original message.

MQRO_EXPIRATION_WITH_FULL_DATA
Report message expiration with full original message.

MQRO_NEW_MSG_ID
Generate a new MSG ID for report message.

MQRO_PASS_CORREL_ID
Use CORREL ID of message for report message.

MQRO_PASS_MSG_ID
Use MSG ID of message for report message.

MsgType (MQLONG)
Message type.

This indicates the type of the message. The value must be one of the following:

MQMT_DATAGRAM
Message not requiring a reply.

MQMT_REQUEST
Message requiring a reply.

MQMT_REPLY
Reply to an earlier request message.

MQMT_REPORT
Report message.

There are no application-defined values.

Expiry (MQLONG)
Message lifetime.

A value of MQEI_UNLIMITED indicates that the message does not expire.

Feedback (MQLONG)
Feedback or reason code.

Feedback codes are grouped as follows:

MQFB_NONE
No feedback provided.

MQFB_SYSTEM_FIRST
Lowest value for system-generated feedback.

MQFB_SYSTEM_LAST
Highest value for system-generated feedback.

MQFB_APPL_FIRST
Lowest value for application-generated feedback.

MQFB_APPL_LAST
Highest value for application-generated feedback.

Applications that generate report messages should not use feedback codes in the system range (other than MQFB_QUIT), unless they wish to simulate report messages generated by the queue manager or message channel agent.

A special feedback code is:

MQFB_QUIT
Application should end.

CodedCharSetId (MQLONG)
Coded character set identifier.

The following value only is defined:

MQCCSI_Q_MGR
Queue manager's coded character set identifier.

Format (MQCHAR8)
Format name.

The queue manager built-in formats are:

MQFMT_NONE
No format name.

MQFMT_ADMIN
Command server request/reply message.

MQFMT_CICS
CICS information header.

MQFMT_COMMAND_1
Type 1 command reply message.

MQFMT_COMMAND_2
Type 2 command reply message.

MQFMT_DEAD_LETTER_HEADER
Dead-letter header.

MQFMT_DIST_HEADER
Distribution-list header.

MQFMT_EVENT
Event message.

MQFMT_IMS
IMS information header.

MQFMT_IMS_VAR_STRING
IMS variable string.

MQFMT_PCF
User-defined message in programmable command format (PCF).

MQFMT_REF_MSG_HEADER
Reference message header.

MQFMT_RF_HEADER
Reference header.

MQFMT_RF_HEADER_2
Reference header format 2.

MQFMT_SAP
SAP information header.

MQFMT_STRING
String.

MQFMT_TRIGGER
Trigger.

Priority (MQLONG)
Message priority.

There is no special value for this field.

Persistence (MQLONG)
Message persistence.

For the MQPUT and MQPUT1 calls, the value must be one of the following:

MQPER_PERSISTENT
Message is persistent.

MQPER_NOT_PERSISTENT
Message is not persistent.

Note: Non-persistent messages are not supported by MQSeries for VSE/ESA. Instead, MQ/VSE propagates the persistence flag on the assumption that the message is destined for a non-VSE/ESA MQ system. Non-persistent messages destined for a VSE/ESA MQ system are treated the same as persistent messages.

MsgId (MQBYTE24)
Message identifier.

The following special value may be used:

MQMI_NONE
No message identifier is specified. A unique identifier will be generated by the queue manager when the message is created.

CorrelId (MQBYTE24)
Correlation identifier.

The following special value may be used:

MQCI_NONE
No correlation identifier is specified.

BackoutCount (MQLONG)
Backout counter.

This is a reserved field.

UserIdentifier (MQCHAR12)
User identifier.

This is a reserved field.

AccountingToken (MQBYTE32)
Accounting token.

This is a reserved field.

ApplIdentityData (MQCHAR32)
Application data relating to identity.

This is a reserved field.

PutApplType (MQLONG)
Type of application that put the message.

This is a reserved field.

PutApplName (MQCHAR28)
Name of application that put the message.

This is a reserved field.

PutDate (MQCHAR8)
Date when message was put.

The format used for the date when this field is generated by the queue manager is:

```
YYYYMMDD
```

where the characters represent (in order):

YYYY	year (four numeric digits)
MM	month of year (01 through 12)
DD	day of month (01 through 31)

Greenwich Mean Time (GMT) is used for the PutDate and PutTime fields, subject to the system clock being set accurately to GMT.

PutTime (MQCHAR8)
Time when message was put. The format used for the time when this field is generated by the queue manager is:

```
HHMMSSTH
```

where the characters represent (in order):

HH	hours (00 through 23)
MM	minutes (00 through 59)
SS	seconds (00 through 59; see note below)
T	tenths of a second (0 through 9)
H	hundredths of a second (0 through 9)

Greenwich Mean Time (GMT) is used for the PutDate and PutTime fields, subject to the system clock being set accurately to GMT.

ApplOriginData (MQCHAR4)
Application data relating to origin.

This is a reserved field.

MQOD – Object descriptor

The MQOD structure is used to specify a queue object. The fields in the Version 1 structure only are supported.

Fields

Version (MQLONG)
Structure version number.

The value must be:

MQOD_VERSION_1
: Version-1 object descriptor structure.

ObjectType (MQLONG)
: Object type.

  Type of object being named in *ObjectName*. The value must be one of the following:

  MQOT_Q_MGR
  : Queue manager.

  MQOT_Q
  : Queue.

DynamicQName (MQCHAR48)
: Dynamic queue name.

  This is a reserved field.

AlternateUserId (MQCHAR12)
: Alternate user identifier.

  This is a reserved field.

MQPMO – Put message options

The fields in the Version 1 structure only are supported.

Fields

Version (MQLONG)
: Structure version number.

  The value must be:

  MQPMO_VERSION_1
  : Version-1 put-message options structure.

Options (MQLONG)
: Options that control the action of MQPUT and MQPUT1.

  The following option only is supported:
  MQPMO_SYNCPOINT

Context (MQHOBJ)
: Object handle of input queue.

  This is a reserved field.

KnownDestCount (MQLONG)
: Number of messages sent successfully to local queues.

  This is a reserved field.

UnknownDestCount (MQLONG)
: Number of messages sent successfully to remote queues.

  This is a reserved field.

InvalidDestCount (MQLONG)
: Number of messages that could not be sent.

  This is a reserved field.

ResolvedQName (MQCHAR48)
Resolved name of destination queue.

This is a reserved field.

MQTM – Trigger message

On the MQSeries for VSE/ESA platform, triggers are invoked by the queue manager using either the transaction ID code, or program ID code, in the queue definition.

The transaction ID, or program ID, determines if the trigger is invoked using an EXEC CICS START, or EXEC CICS LINK, program respectively.

Trigger programs using the START mechanism can use the EXEC CICS RETRIEVE program to retrieve the trigger data structure. Programs invoked by the LINK mechanism can retrieve the MQTM structure in the DFHCOMMAREA.

Fields

Version (MQLONG)
Structure version number.

The value must be:

MQTM_VERSION_1
Version number for trigger message structure.

ProcessName (MQCHAR48)
Name of process object.

This is a reserved field.

TriggerData (MQCHAR64)
Trigger data.

On MQSeries for VSE/ESA this is a 13-byte field, consisting of:
- 4-byte transaction ID code
- 8-byte program ID code
- 1-byte trigger-event flag

The trigger event flag indicates whether the trigger was started from a (F)irst message or (E)very message event. When a trigger instance is a program (rather than a transaction), the MQTM data structure is passed to the trigger program as a COMMAREA.

In the case of (E)very event triggers, the queue manager will start another trigger instance when a trigger instance ends, if it detects that there are still messages on the queue.

If an error has occurred such that the trigger program cannot process messages from the object queue, a loop condition may arise (the queue manager will continue to start trigger instances which themselves continue to fail).

This can be avoided by setting the trigger event flag in the TriggerData field of the MQTM data structure to “S” (stop). Since the trigger event flag is in a COMMAREA, the queue manager will detect that the event flag has been set to (S)top, and will cease starting new trigger instances.

ApplType (MQLONG)
Application type.

On MQSeries for VSE/ESA *ApplType* has the following standard value:
MQAT_CICS_VSE
MQSeries for VSE/ESA application.

The initial value of this field is `0`.

ApplId (MQCHAR256)
Application identifier.

On MQSeries for VSE/ESA *ApplId* is:
- A CICS transaction ID (for MQAT_CICS_VSE applications).

EnvData (MQCHAR128)
Environment data.

This is a reserved field.

UserData (MQCHAR128)
User data.

Contains user-defined data configured in the queue definition of the queue that initiates the trigger instance.

MQI calls

This section identifies the MQI calls supported by MQSeries for VSE/ESA, which are:
MQBACK
Back out changes
MQCLOSE
Close object
MQCMIT
Commit changes
MQCONN
Connect queue manager
MQDISC
Disconnect queue manager
MQGET
Get message
MQINQ
Inquire about object attributes
MQOPEN
Open object
MQPUT
Put message
MQPUT1
Put one message
MQSET
Set object attributes

MQBACK – Back out changes

The MQBACK call indicates to the queue manager that all of the message gets and puts that have occurred since the last syncpoint are to be backed out (for client and batch programs) or have been backed out (for online programs). Messages put as part of a unit of work are deleted; messages retrieved as part of a unit of work are reinstated on the queue (for client and batch programs).

Note: For online applications, it is required that the application itself performs a CICS SYNCPOINT ROLLBACK before issuing the MQBACK call. The MQBACK call is used to re-enter the queue manager to update internal MQ/VSE tables.

For client and batch programs, the CICS SYNCPOINT ROLLBACK is performed for the caller.

Syntax

```
MQBACK (Hconn,CompCode,Reason)
```

Parameters

The MQBACK call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

CompCode (MQLONG) – output
Completion code.

It is one of the following:

`MQCC_OK`
Successful completion.

`MQCC_FAILED`
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

`MQRC_NONE`
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQBACK (Hconn,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQBACK' USING HCONN,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQBACK (HCONN,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN       FIXED BIN(31); /*Connection handle */
DCL COMPCODE    FIXED BIN(31); /*Completion code */
DCL REASON      FIXED BIN(31); /*Reason code qualifying CompCode */
```

MQCLOSE – Close object

The MQCLOSE call relinquishes access to an object, and is the inverse of the MQOPEN call.

Syntax

```
MQCLOSE (Hconn,Hobj,Options,CompCode,Reason)
```

Parameters

The MQCLOSE call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

Hobj (MQHOBJ) – input/output
Object handle.

This handle represents the object that is being closed. The object can be of any type. The value of Hobj was returned by a previous MQOPEN call. On successful completion of the call, the queue manager sets this parameter to a value of binary zeroes.

Options (MQLONG) – input
Options that control the action of MQCLOSE.

The following option only is supported, and must be specified:

MQCO_NONE
No optional close processing required.

CompCode (MQLONG) – output
Completion code.

It is one of the following:

MQCC_OK
Successful completion.

MQCC_WARNING
Warning (partial completion).

MQCC_FAILED
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.

MQRC_HOBJ_ERROR
(2019, X'7E3') Object handle not valid.
MQRC_OPTIONS_ERROR
(2046, X'7FE') Options not valid or not consistent.
MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQCLOSE (Hconn,&Hobj,Options,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQHOBJ  Hobj;           /*Object handle */
MQLONG  Options;        /*Options that control the action of MQCLOSE */
MQLONG  CompCode;       /*Completion code */
MQLONG  Reason;         /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQCLOSE'USING HCONN,HOBJ,OPTIONS,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object handle
  01 HOBJ PICS9(9)BINARY.
**Options that control the action of MQCLOSE
  01 OPTIONS PICS9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQCLOSE (HCONN,HOBJ,OPTIONS,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN       FIXED BIN(31);  /*Connection handle */
DCL HOBJ        FIXED BIN(31);  /*Object handle */
DCL OPTIONS     FIXED BIN(31);  /*Options that control the action of
                                  MQCLOSE */
DCL COMPCODE    FIXED BIN(31);  /*Completion code */
DCL REASON      FIXED BIN(31);  /*Reason code qualifying CompCode */
```

MQCMIT – Commit changes

For client and batch programs, the MQCMIT call indicates to the queue manager that the application has reached a syncpoint, and that all of the message gets and puts that have occurred since the last syncpoint are to be made permanent by the queue manager issuing a CICS SYNCPOINT for the application. Messages put as part of a unit of work are made available to other applications; messages retrieved as part of a unit of work are deleted.

For online application programs, the MQCMIT call (issued after the online application issues a CICS SYNCPOINT) indicates to the queue manager it can verify that syncpoint has occurred, update internal tables and clear any delayed gets.

Syntax

```
MQCMIT (Hconn,CompCode,Reason)
```

Parameters

The MQCMIT call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

CompCode (MQLONG) – output
Completion code.

It is one of the following:

MQCC_OK
Successful completion.

MQCC_FAILED
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQCMIT (Hconn,&CompCode,&Reason)
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQCMIT'USING HCONN,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQCMIT (HCONN,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN       FIXED BIN(31);  /*Connection handle */
DCL COMPCODE    FIXED BIN(31);  /*Completion code */
DCL REASON      FIXED BIN(31);  /*Reason code qualifying CompCode */
```

MQCONN – Connect queue manager

The MQCONN call connects an application program to a queue manager. It provides a queue manager connection handle, which is used by the application on subsequent message queuing calls.

Syntax

```
MQCONN (QMgrName,Hconn,CompCode,Reason)
```

Parameters

The MQCONN call has the following parameters:

QMgrName (MQCHAR48) – input
Name of queue manager.

The name specified must be the name of a local queue manager. In MQSeries for VSE/ESA there is only one queue manager in a CICS partition.

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. It must be specified on all subsequent message queuing calls issued by the application. It ceases to be valid when the MQDISC call is issued, or when the CICS transaction terminates.

CompCode (MQLONG) – output
Completion code.

It is one of the following:

MQCC_OK
: Successful completion.

MQCC_FAILED
: Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
: (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_MAX_CONNS_LIMIT_REACHED
: (2025, X'7E9') Maximum number of connections reached.

MQRC_Q_MGR_NAME_ERROR
: (2058, X'80A') Queue manager name not valid or not known.

MQRC_Q_MGR_NOT_AVAILABLE
: (2059, X'80B') Queue manager not available for connection.

MQRC_STORAGE_NOT_AVAILABLE
: (2071, X'817') Insufficient storage available.

MQRC_UNEXPECTED_ERROR
: (2195, X'893') Unexpected error occurred.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQCONN (QMgrName,&Hconn,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQCHAR48 QMgrName;      /*Name of queue manager */
MQHCONN Hconn;          /*Connection handle */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQCONN'USING QMGRNAME,HCONN,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Name of queue manager
  01 QMGRNAME PICX(48).
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQCONN (QMGRNAME,HCONN,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL QMGRNAME    CHAR(48);       /*Name of queue manager */
DCL HCONN       FIXED BIN(31);  /*Connection handle */
DCL COMPCODE    FIXED BIN(31);  /*Completion code */
DCL REASON      FIXED BIN(31);  /*Reason code qualifying CompCode */
```

MQDISC – Disconnect queue manager

The MQDISC call breaks the connection between the queue manager and the application program. It is the inverse of the MQCONN call.

Syntax

```
MQDISC (Hconn,CompCode,Reason)
```

Parameters

The MQDISC call has the following parameters:

Hconn (MQHCONN) – input/output
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

On successful completion of the call, the queue manager sets Hconn to binary zeroes.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_FAILED:
MQRC_ADAPTER_DISC_LOAD_ERROR
(2138, X'85A') Unable to load adapter disconnection module.
MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.
MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.
MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQDISC (&Hconn,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQDISC'USING HCONN,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN    PIC S9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON   PIC S9(9)BINARY.
```

PL/I invocation

```
CALL MQDISC (HCONN,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN       FIXED BIN(31);  /*Connection handle */
DCL COMPCODE    FIXED BIN(31);  /*Completion code */
DCL REASON      FIXED BIN(31);  /*Reason code qualifying CompCode */
```

MQGET – Get message

The MQGET call retrieves a message from a local queue that has been opened using the MQOPEN call.

Syntax

```
MQGET (Hconn,Hobj,MsgDesc,GetMsgOpts,BufferLength,
Buffer,DataLength,CompCode,Reason)
```

Parameters

The MQGET call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

Hobj (MQHOBJ) – input
Object handle.

This handle represents the queue from which a message is to be retrieved. The value of Hobj was returned by a previous MQOPEN call. The queue must have been opened with one or more of the following options (see "MQOPEN – Open object" on page 339 for details):
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_BROWSE

MsgDesc (MQMD) – input/output
Message descriptor.

This structure describes the attributes of the message required, and the attributes of the message retrieved. See "MQMD – Message descriptor" on page 314 for details.

If BufferLength is less than the message length, MsgDesc is still filled in by the queue manager, whether or not MQGMO_ACCEPT_TRUNCATED_MSG is

specified on the GetMsgOpts parameter (see the Options field described in "MQGMO – Get message options" on page 313.

GetMsgOpts (MQGMO) – input/output
Options that control the action of MQGET. See "MQGMO – Get message options" on page 313 for details.

BufferLength (MQLONG) – input
Length in bytes of the Buffer area.

Zero can be specified for messages that have no data, or if the message is to be removed from the queue and the data discarded (MQGMO_ACCEPT_TRUNCATED_MSG must be specified in this case).

Buffer (MQBYTE×BufferLength) – output
Area to contain the message data.

If BufferLength is less than the message length, as much of the message as possible is moved into Buffer ; this happens whether or not MQGMO_ACCEPT_TRUNCATED_MSG is specified on the GetMsgOpts parameter (see the Options field described in "MQGMO – Get message options" on page 313).

The character set and encoding of the data in Buffer are given (respectively) by the CodedCharSetId and Encoding fields returned in the MsgDesc parameter. If these are different from the values required by the receiver, the receiver must convert the application message data to the character set and encoding required. The MQGMO_CONVERT option can be used with a user-written exit to perform the conversion of the message data (see "MQGMO – Get message options" on page 313 for details of this option).

Note: All of the other parameters on the MQGET call are in the character set of the local queue manager.

DataLength (MQLONG) – output
Length of the message.

This is the length in bytes of the application data in the message. If this is greater than BufferLength , only BufferLength bytes are returned in the Buffer parameter (that is, the message is truncated). If the value is zero, it means that the message contains no application data.

If BufferLength is less than the message length, DataLength is still filled in by the queue manager, whether or not MQGMO_ACCEPT_TRUNCATED_MSG is specified on the GetMsgOpts parameter (see the Options field described in "MQGMO – Get message options" on page 313 for more information).

This allows the application to determine the size of the buffer required to accommodate the message data, and then reissue the call with a buffer of the appropriate size.

However, if the MQGMO_CONVERT option is specified, and the converted message data is too long to fit in Buffer , the value returned for DataLength is:

- The length of the unconverted data, for queue-manager defined formats. In this case, if the nature of the data causes it to expand during conversion, the application must allocate a buffer somewhat bigger than the value returned by the queue manager for DataLength.
- The value returned by the data-conversion exit, for application-defined formats.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

The reason codes listed below are the ones that the queue manager can return for the *Reason* parameter. If the application specifies the MQGMO_CONVERT option, and a user-written exit is invoked to convert some or all of the message data, it is the exit that decides what value is returned for the *Reason* parameter. As a result, values other than those documented below are possible.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_WARNING:
MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Truncated message returned (processing completed).
MQRC_TRUNCATED_MSG_FAILED
(2080, X'820') Truncated message returned (processing not completed).

If *CompCode* is MQCC_FAILED:
MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Buffer length parameter not valid.
MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Converted data too big for buffer.
MQRC_DBCS_ERROR
(2150, X'866') DBCS string not valid.
MQRC_FILE_SYSTEM_ERROR
(2216, X'8A8') Queuer received file error.
MQRC_FORMAT_ERROR
(2110, X'83E') Format field not valid.
MQRC_GET_INHIBITED
(2016, X'7E0') Gets inhibited for the queue.
MQRC_GMO_ERROR
(2186, X'88A') Get-message options structure not valid.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.
MQRC_HOBJ_ERROR
(2019, X'7E3') Object handle not valid.
MQRC_MD_ERROR
(2026, X'7EA') Message descriptor not valid.
MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') No message available.
MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') Browse cursor not positioned on message.
MQRC_NOT_CONVERTED
(2119, X'847') Application message data not converted.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Queue not open for browse.
MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Queue not open for input.
MQRC_OPTIONS_ERROR
(2046, X'7FE') Options not valid or not consistent.
MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Source coded character set identifier not valid.
MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Packed-decimal encoding in message not recognized.
MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Floating-point encoding in message not recognized.
MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Source integer encoding not recognized.
MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Source length parameter not valid.
MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.
MQRC_TARGET_CCSID_ERROR
(2115, X'843') Target coded character set identifier not valid.
MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Packed-decimal encoding specified by receiver not recognized.
MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Floating-point encoding specified by receiver not recognized.
MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Target integer encoding not recognized.
MQRC_TARGET_LENGTH_ERROR
(2144, X'860') Target length parameter not valid.
MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') Wait interval in MQGMO not valid.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQGET (Hconn,Hobj,&MsgDesc,&GetMsgOpts,BufferLength,Buffer,
&DataLength,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQHOBJ Hobj;            /*Object handle */
MQMD MsgDesc;           /*Message descriptor */
MQGMO GetMsgOpts;       /*Options that control the action of MQGET */
MQLONG BufferLength;    /*Length in bytes of the Buffer area */
MQBYTE Buffer[n];       /*Area to contain the message data */
MQLONG DataLength;      /*Length of the message */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQGET'USING HCONN,HOBJ,MSGDESC,GETMSGOPTS,
BUFFERLENGTH,BUFFER,DATALENGTH,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object handle
  01 HOBJ PIC  S9(9)BINARY.
**Message descriptor
```

```
  01 MSGDESC.
      COPY CMQMDV.
**Options that control the action of MQGET
  01 GETMSGOPTS.
      COPY CMQGMOV.
**Length in bytes of the Buffer area
  01 BUFFERLENGTH PIC S9(9)BINARY.
**Area to contain the message data
  01 BUFFER PIC X(n).
**Length of the message
  01 DATALENGTH PIC S9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PIC S9(9)BINARY.
```

PL/I invocation

```
CALL MQGET (HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,BUFFER,
DATALENGTH,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN       FIXED BIN(31); /*Connection handle */
DCL HOBJ        FIXED BIN(31); /*Object handle */
DCL MSGDESC     LIKE MQMD;     /*Message descriptor */
DCL GETMSGOPTS  LIKE MQGMO;    /*Options that control the action of MQGET */
DCL BUFFERLENGTH FIXED BIN(31);/*Length in bytes of the Buffer area */
DCL BUFFER      CHAR(n);       /*Area to contain the message data */
DCL DATALENGTH  FIXED BIN(31); /*Length of the message */
DCL COMPCODE    FIXED BIN(31); /*Completion code */
DCL REASON      FIXED BIN(31); /*Reason code qualifying CompCode */
```

MQINQ – Inquire about object attributes

The MQINQ call returns an array of integers and a set of character strings containing the attributes of an object. The following types of object are valid:

- Queue
- Queue manager

Syntax

```
MQINQ (Hconn,Hobj,SelectorCount,Selectors,IntAttrCount,
IntAttrs,CharAttrLength,CharAttrs,CompCode,Reason)
```

Parameters

The MQINQ call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

Hobj (MQHOBJ) – input
Object handle.

This handle represents the object (of any type) whose attributes are required. The handle must have been returned by a previous MQOPEN call that specified the MQOO_INQUIRE option.

SelectorCount (MQLONG) – input
Count of selectors.

This is the count of selectors that are supplied in the Selectors array. It is the number of attributes that are to be returned. Zero is a valid value. The maximum number allowed is 256.

Selectors (MQLONG×*SelectorCount*) – input
Array of attribute selectors.

This is an array of SelectorCount attribute selectors; each selector identifies an attribute (integer or character) whose value is required. Each selector must be valid for the type of object that Hobj represents, otherwise the call fails with completion code MQCC_FAILED and reason code MQRC_SELECTOR_ERROR.

In the special case of queues:

- If the selector is not valid for queues of any type, the call fails with completion code MQCC_FAILED and reason code MQRC_SELECTOR_ERROR.
- If the selector is applicable only to queues of type or types other than that of the object, the call succeeds with completion code MQCC_WARNING and reason code MQRC_SELECTOR_NOT_FOR_TYPE.

Selectors for queue managers:

MQCA_ALTERATION_DATE
: Last modification date (Length of 12, format 'YYYY-MM-DD ').

MQCA_ALTERATION_TIME
: Last modification time (Length of 8, format 'HH:MM:SS').

MQCA_BATCH_INTERFACE_ID
: Batch interface identifier (MQ_BATCH_INTERFACE_ID_LENGTH).

MQCA_COMMAND_INPUT_Q_NAME
: System command queue name (MQ_Q_NAME_LENGTH).

MQCA_COMMAND_REPLY_Q_NAME
: MQSC reply queue name (MQ_Q_NAME_LENGTH).

MQCA_DEAD_LETTER_Q_NAME
: System dead letter queue name (MQ_Q_NAME_LENGTH).

MQCA_MONITOR_Q_NAME
: MQI monitor queue name (MQ_Q_NAME_LENGTH).

MQCA_Q_MGR_DESC
: Queue manager description (MQ_Q_MGR_DESC_LENGTH).

MQCA_Q_MGR_NAME
: Queue manager name (MQ_Q_MGR_NAME_LENGTH).

MQCA_SSL_KEY_LIBRARY
: SSL key library name (MQ_SSL_KEY_LIBRARY_LENGTH).

MQCA_SSL_KEY_MEMBER
: SSL key library member name (MQ_SSL_KEY_MEMBER_LENGTH).

MQCA_SYSTEM_LOG_Q_NAME
: System log queue name (MQ_Q_NAME_LENGTH).

MQIA_BATCH_INTERFACE_AUTO
: Indicator for the automatic activation of the batch interface. Can be one of the following values:
  MQAUTO_START_NO
  MQAUTO_START_YES

MQIA_CMD_SERVER_AUTO
: Indicator for the automatic activation of the PCF command server. Can be one of the following values:
  MQAUTO_START_NO
  MQAUTO_START_YES

MQIA_CMD_SERVER_CONVERT_MSG
: Indicator for the data conversion of PCF messages. Can be one of the following values:
  MQCSRV_CONVERT_NO

MQCSRV_CONVERT_YES

MQIA_CMD_SERVER_DLQ_MSG
Indicator for the storage of undeliverable PCF reply messages. Can be one of the following values:
MQCSRV_DLQ_NO
MQCSRV_DLQ_YES

MQIA_CODED_CHAR_SET_ID
Local code page for queue manager.

MQIA_COMMAND_LEVEL
Supported command level. For MQSeries for VSE/ESA, this value is always MQCMDL_LEVEL_211.

MQIA_DIST_LISTS
Indicator for distributed list support. For MQSeries for VSE/ESA, this value is always MQDL_NOT_SUPPORTED.

MQIA_LISTENER_PORT_NUMBER
Port number for TCP/IP Listener process.

MQIA_MAX_GLOBAL_LOCKS
Buffer size for queue manager to manage concurrent queue access.

MQIA_MAX_HANDLES
Maximum number of concurrent connections to the queue manager.

MQIA_MAX_LOCAL_LOCKS
Buffer size for applications to manage concurrent queue access.

MQIA_MAX_MSG_LENGTH
Maximum message length for queue messages.

MQIA_MAX_OPEN_Q
Maximum number of concurrently open queues.

MQIA_MAX_Q_DEPTH
Maximum allowable queue depth for queues.

MQIA_MONITOR_INTERVAL
Queue manager housekeeping process interval.

MQIA_PLATFORM
MQSeries system platform identifier. For MQSeries for VSE/ESA, this value is always MQPL_VSE.

MQIA_Q_USERS
Maximum number of concurrent opens per queue.

MQIA_SYNCPOINT
Indicator for SYNCPOINT support. For MQSeries for VSE/ESA, this value is always MQSP_NOT_AVAILABLE.

| MQIA_AUTHORITY_EVENT
| Control attribute for authority events.
| MQIA_INHIBIT_EVENT
| Control attribute for inhibit events.
| MQIA_LOCAL_EVENT
| Control attribute for local events.
| MQIA_REMOTE_EVENT
| Control attribute for remote events.
| MQIA_START_STOP_EVENT
| Control attribute for start stop events.
| MQIA_PERFORMANCE_EVENT
| Control attribute for performance events.

Selectors for all types of queue:

MQCA_CREATION_DATE
Queue creation date (Length of 12, format 'YYYY-MM-DD ').

MQCA_CREATION_TIME
Queue creation time (Length of 8, format 'HH:MM:DD').
MQCA_Q_DESC
Queue description (MQ_Q_DESC_LENGTH).
MQCA_Q_NAME
Queue name (MQ_Q_NAME_LENGTH).
MQIA_INHIBIT_PUT
Whether put operations are allowed. Can be one of the following values:
MQQA_PUT_ALLOWED
MQQA_PUT_INHIBITED
MQIA_Q_TYPE
Queue type. Can be one of the following values:
MQQT_ALIAS
MQQT_LOCAL
MQQT_REMOTE

Selectors for local queues:
MQCA_CICS_FILE_NAME
CSD file name for queue messages (MQ_CICS_FILE_NAME_LENGTH).
MQCA_TRIGGER_CHANNEL_NAME
Channel name for MCA trigger process (MQ_CHANNEL_NAME_LENGTH).
MQCA_TRIGGER_DATA
Trigger user data (MQ_PROCESS_USER_DATA_LENGTH).
MQCA_TRIGGER_PROGRAM_NAME
Program name for trigger process (MQ_TRIGGER_PROGRAM_NAME_LENGTH).
MQCA_TRIGGER_TERM_ID
Terminal identifier for trigger process (MQ_TRIGGER_TERM_ID_LENGTH).
MQCA_TRIGGER_TRANS_ID
Transaction identifier for trigger process (MQ_TRIGGER_TRANS_ID_LENGTH).
MQIA_CURRENT_Q_DEPTH
Current queue depth.
MQIA_DEF_PERSISTENCE
Default persistence for queue. For MQSeries for VSE/ESA, this value is always MQPER_PERSISTENT.
MQIA_DEFINITION_TYPE
Queue definition type. For MQSeries for VSE/ESA, this value is always MQQDT_PREDEFINED.
MQIA_INHIBIT_GET
Whether get operations are allowed. Can be one of the following values:
MQQA_GET_ALLOWED
MQQA_GET_INHIBITED
MQIA_MAX_GLOBAL_LOCKS
Buffer size for queue manager to manage concurrent queue access.
MQIA_MAX_LOCAL_LOCKS
Buffer size for applications to manage concurrent queue access.
MQIA_MAX_MSG_LENGTH
Maximum message length for queue messages.

MQIA_MAX_Q_DEPTH
Maximum allowable queue depth for queues.
MQIA_MAX_Q_TRIGGERS
Maximum number of concurrent trigger instances for a particular queue.
MQIA_OPEN_INPUT_COUNT
Number of opens for input currently issued against queue.
MQIA_OPEN_OUTPUT_COUNT
Number of opens for output currently issued against queue.
MQIA_Q_USERS
Maximum number of concurrent opens per queue.
MQIA_SHAREABILITY
Queue shareability mode. Can be one of the following values:
MQQA_SHAREABLE
MQQA_NOT_SHAREABLE
MQIA_TRIGGER_CONTROL
Whether a trigger is required for the queue. Can be one of the following values:
MQTC_OFF
MQTC_ON
MQIA_TRIGGER_RESTART
Indicator for the reactivation of a trigger process. Can be one of the following values:
MQTRIGGER_RESTART_YES
MQTRIGGER_RESTART_NO
MQIA_TRIGGER_TYPE
Trigger event type. Can be one of the following values:
MQTT_NONE
MQTT_FIRST
MQTT_EVERY
MQIA_USAGE
Queue usage. Can be one of the following values:
MQUS_NORMAL
MQUS_TRANSMISSION
MQIA_Q_DEPTH_HIGH_LIMIT
High limit for queue depth.
MQIA_Q_DEPTH_LOW_LIMIT
Low limit for queue depth.
MQIA_Q_DEPTH_MAX_EVENT
Control attribute for queue depth max events.
MQIA_Q_DEPTH_HIGH_EVENT
Control attribute for queue depth high events.
MQIA_Q_DEPTH_LOW_EVENT
Control attribute for queue depth low events.
MQIA_Q_SERVICE_INTERVAL
Limit for queue service interval.
MQIA_Q_SERVICE_INTERVAL_EVENT
Control attribute for queue service interval events.

Selectors for local definitions of remote queues
MQCA_REMOTE_Q_MGR_NAME
Name of remote queue manager (MQ_Q_MGR_NAME_LENGTH).
MQCA_REMOTE_Q_NAME
Name of remote queue as known on remote queue manager (MQ_Q_NAME_LENGTH).

MQCA_XMIT_Q_NAME
Name of local transmission queue.

Selectors for alias queues
MQCA_BASE_Q_NAME
Name of queue that alias resolves to (MQ_Q_NAME_LENGTH).
MQIA_INHIBIT_GET
Whether get operations are allowed.

IntAttrCount (MQLONG) – input
Count of integer attributes.

This is the number of elements in the IntAttrs array. Zero is a valid value. If this is at least the number of MQIA_* selectors in the Selectors parameter, all integer attributes requested are returned.

IntAttrs (MQLONG×IntAttrCount) – output
Array of integer attributes.

This is an array of IntAttrCount integer attribute values.

Integer attribute values are returned in the same order as the MQIA_* selectors in the Selectors parameter. If the array contains more elements than the number of MQIA_* selectors, the excess elements are unchanged.

If Hobj represents a queue, but an attribute selector is not applicable to that type of queue, the specific value MQIAV_NOT_APPLICABLE is returned for the corresponding element in the IntAttrs array.

If the IntAttrCount or SelectorCount parameter is zero, IntAttrs is not referred to; in this case, the parameter address passed by programs written in C or System/390 assembler may be null.

CharAttrLength (MQLONG) – input
Length of character attributes buffer.

This is the length in bytes of the CharAttrs parameter.

This must be at least the sum of the lengths of the requested character attributes (see Selectors). Zero is a valid value.

CharAttrs (MQCHAR×CharAttrLength) – output
Character attributes.

This is the buffer in which the character attributes are returned, concatenated together. The length of the buffer is given by the CharAttrLength parameter. Character attributes are returned in the same order as the MQCA_* selectors in the Selectors parameter. The length of each attribute string is fixed for each attribute (see Selectors), and the value in it is padded to the right with blanks if necessary. If the buffer is larger than that needed to contain all of the requested character attributes (including padding), the bytes beyond the last attribute value returned are unchanged.

If Hobj represents a queue, but an attribute selector is not applicable to that type of queue, a character string consisting entirely of asterisks (*) is returned as the value of that attribute in CharAttrs .

If the CharAttrLength or SelectorCount parameter is zero, CharAttrs is not referred to; in this case, the parameter address passed by programs written in C may be null.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
MQCC_OK
: Successful completion.
MQCC_FAILED
: Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
: (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_CHAR_ATTR_LENGTH_ERROR
: (2006, X'7D6') Length of character attributes not valid.

MQRC_CHAR_ATTRS_ERROR
: (2007, X'7D7') Character attributes string not valid.

MQRC_CONNECTION_BROKEN
: (2009, X'7D9') Connection to queue manager lost.

MQRC_HCONN_ERROR
: (2018, X'7E2') Connection handle not valid.

MQRC_HOBJ_ERROR
: (2019, X'7E3') Object handle not valid.

MQRC_INT_ATTR_COUNT_ERROR
: (2021, X'7E5') Count of integer attributes not valid.

MQRC_INT_ATTRS_ARRAY_ERROR
: (2023, X'7E7') Integer attributes array not valid.

MQRC_NOT_OPEN_FOR_INQUIRE
: (2038, X'7F6') Queue not open for inquire.

MQRC_SELECTOR_COUNT_ERROR
: (2065, X'811') Count of selectors not valid.

MQRC_SELECTOR_ERROR
: (2067, X'813') Attribute selector not valid.

MQRC_SELECTOR_LIMIT_EXCEEDED
: (2066, X'812') Count of selectors too big.

MQRC_STORAGE_NOT_AVAILABLE
: (2071, X'817') Insufficient storage available.

MQRC_UNEXPECTED_ERROR
: (2195, X'893') Unexpected error occurred.

Language invocations

This call is supported in the following programming languages:
C invocation

```
MQINQ (Hconn,Hobj,SelectorCount,Selectors,IntAttrCount,IntAttrs,
CharAttrLength,CharAttrs,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQHOBJ Hobj;            /*Object handle */
MQLONG SelectorCount;   /*Count of selectors */
MQLONG Selectors [n];   /*Array of attribute selectors */
MQLONG IntAttrCount;    /*Count of integer attributes */
MQLONG IntAttrs [n];    /*Array of integer attributes */
MQLONG CharAttrLength;  /*Length of character attributes buffer */
MQCHAR CharAttrs [n];   /*Character attributes */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQINQ'USING HCONN,HOBJ,SELECTORCOUNT, SELECTORS-
TABLE,INTATTRCOUNT,INTATTRS-TABLE,
CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object handle
  01 HOBJ PICS9(9)BINARY.
**Count of selectors
  01 SELECTORCOUNT PIC S9(9)BINARY.
**Array of attribute selectors
  01 SELECTORS-TABLE.
     02 SELECTORS PIC S9(9)BINARY OCCURS n TIMES.
**Count of integer attributes
  01 INTATTRCOUNT PIC S9(9)BINARY.
**Array of integer attributes
  01 INTATTRS-TABLE.
     02 INTATTRS PIC S9(9)BINARY OCCURS n TIMES.
**Length of character attributes buffer
  01 CHARATTRLENGTH PIC S9(9)BINARY.
**Character attributes
  01 CHARATTRS PIC X(n).
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQINQ (HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,
INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN FIXED    BIN(31);        /*Connection handle */
DCL HOBJ FIXED     BIN(31);        /*Object handle */
DCL SELECTORCOUNT  FIXED BIN(31);  /*Count of selectors */
DCL SELECTORS(N)   FIXED BIN(31);  /*Array of attribute selectors */
DCL INTATTRCOUNT   FIXED BIN(31);  /*Count of integer attributes */
DCL INTATTRS(N)    FIXED BIN(31);  /*Array of integer attributes */
DCL CHARATTRLENGTH FIXED BIN(31);  /*Length of character attributes buffer */
DCL CHARATTRS      CHAR(N);        /*Character attributes */
DCL COMPCODE       FIXED BIN(31);  /*Completion code */
DCL REASON         FIXED BIN(31);  /*Reason code qualifying CompCode */
```

MQOPEN – Open object

The MQOPEN call establishes access to an object. The following types of object are valid:

Queue
Queue manager

Syntax

```
MQOPEN (Hconn,ObjDesc,Options,Hobj,CompCode,Reason)
```

Parameters

The MQOPEN call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

ObjDesc (MQOD) – input/output
Object descriptor.

This is a structure that identifies the object to be opened; see "MQOD – Object descriptor" on page 318 for details.

Options (MQLONG) – input
Options that control the action of MQOPEN.

The following options apply and you must specify at least one of these. However, you cannot specify the two input options together and you cannot specify an input option with an output option.

MQOO_BROWSE
MQOO_INPUT_SHARED
MQOO_INPUT_EXCLUSIVE
MQOO_INQUIRE
MQOO_OUTPUT
MQOO_SET

`MQOO_INPUT_SHARED`
Open queue to get messages with shared access.

The queue is opened for use with subsequent MQGET calls. The call can succeed if the queue is currently open by this or another application with MQOO_INPUT_SHARED, but fails with reason code MQRC_OBJECT_IN_USE if the queue is currently open with MQOO_INPUT_EXCLUSIVE.

`MQOO_INPUT_EXCLUSIVE`
Open queue to get messages with exclusive access.

The queue is opened for use with subsequent MQGET calls. The call fails with reason code MQRC_OBJECT_IN_USE if the queue is currently open by this or another application for input of any type (MQOO_INPUT_SHARED or MQOO_INPUT_EXCLUSIVE).

This option is valid only for local, alias, and model queues; it is not valid for remote queues.

The following notes apply to these options:

- Only one of these options can be specified.
- An MQOPEN call with one of these options can succeed even if the InhibitGet queue attribute is set to MQQA_GET_INHIBITED (although subsequent MQGET calls will fail while the attribute is set to this value).
- If an alias queue is opened with one of these options, the test for exclusive use (or for whether another application has exclusive use) is against the base queue to which the alias resolves.
- These options are not valid if ObjectQMgrName is the name of a queue manager alias; this is true even if the value of the RemoteQMgrName attribute in the local definition of a remote queue used for queue-manager aliasing is the name of the local queue manager.

`MQOO_BROWSE`
Open queue to browse messages.

The queue is opened for use with subsequent MQGET calls with one of the following options:

1. MQGMO_BROWSE_FIRST
2. MQGMO_BROWSE_NEXT
3. MQGMO_BROWSE_MSG_UNDER_CURSOR

This is allowed even if the queue is currently open for MQOO_INPUT_EXCLUSIVE. An MQOPEN call with the MQOO_BROWSE option establishes a browse cursor, and positions it logically before the first message on the queue; see "MQGET – Get message" on page 328 for further information.

This option is valid only for local and alias; it is not valid for remote queues and objects which are not queues. It is also not valid if ObjectQMgrName is the name of a queue manager alias; this is true even if the value of the RemoteQMgrName attribute in the local definition of a remote queue used for queue-manager aliasing is the name of the local queue manager.

`MQOO_OUTPUT`
Open queue to put messages.

The queue is opened for use with subsequent MQPUT calls. An MQOPEN call with this option can succeed even if the InhibitPut queue attribute is set to MQQA_PUT_INHIBITED (although subsequent MQPUT calls will fail while the attribute is set to this value).

`MQOO_INQUIRE`
Open object to inquire attributes.

The queue or queue manager is opened for use with subsequent MQINQ calls.

This option is not valid if ObjectQMgrName is the name of a queue manager alias; this is true even if the value of the RemoteQMgrName attribute in the local definition of a remote queue used for queue-manager aliasing is the name of the local queue manager.

`MQOO_SET`
Open queue to set attributes.

The queue is opened for use with subsequent MQSET calls. This option is valid for all queue types supported by MQSeries for VSE/ESA.

Hobj (MQHOBJ) – output
Object handle.

This handle represents the access that has been established to the object. It must be specified on subsequent message queuing calls that operate on the object. It ceases to be valid when the MQCLOSE call is issued, or when the CICS task terminates.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_FAILED:
MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') Alias base queue not a valid type.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') No more handles available.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.
MQRC_OBJECT_IN_USE
(2042, X'7FA') Object already open with conflicting options.
MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Object type not valid.
MQRC_OD_ERROR
(2044, X'7FC') Object descriptor structure not valid.
MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Option not valid for object type.
MQRC_OPTIONS_ERROR
(2046, X'7FE') Options not valid or not consistent.
MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.
MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.
MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822') Unknown alias base queue.
MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.
MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826') Unknown object queue manager.
MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827') Unknown remote queue manager.

Usage notes

1. The object opened is one of the following:
   - A queue, in order to:
     - Get or browse messages (using the MQGET call)
     - Put messages (using the MQPUT call)
     - Inquire about the attributes of the queue (using the MQINQ call)
     - Set the attributes of the queue (using the MQSET call)
   - The queue manager, in order to:
     - Inquire about the attributes of the local queue manager (using the MQINQ call).
2. It is valid for an application to open the same object more than once. A different object handle is returned for each open. Each handle that is returned can be used for the functions for which the corresponding open was performed.
3. If security is enabled, the queue manager performs security checks when an MQOPEN call is issued, to verify that the user identifier under which the application is running has the appropriate level of authority before access is permitted. The authority check is made on the name of the object being opened, and not on the name, or names, resulting after a name has been resolved.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQOPEN (Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQOD ObjDesc;           /*Object descriptor */
MQLONG Options;         /*Options that control the action of MQOPEN */
MQHOBJ Hobj;            /*Object handle */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQOPEN'USING HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object descriptor
  01 OBJDESC.
  COPY CMQODV.
**Options that control the action of MQOPEN
  01 OPTIONS PICS9(9)BINARY.
**Object handle
  01 HOBJ PICS9(9)BINARY.
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQOPEN (HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN       FIXED BIN(31); /*Connection handle */
DCL OBJDESC     LIKE MQOD;     /*Object descriptor */
DCL OPTIONS     FIXED BIN(31); /*Options that control the action of
                                    MQOPEN */
DCL HOBJ        FIXED BIN(31); /*Object handle */
DCL COMPCODE    FIXED BIN(31); /*Completion code */
DCL REASON      FIXED BIN(31); /*Reason code qualifying CompCode */
```

MQPUT – Put message

The MQPUT call puts a message on a queue. The queue must already be open.

Syntax

```
MQPUT (Hconn,Hobj,MsgDesc,PutMsgOpts,BufferLength, Buffer,CompCode,Reason)
```

Parameters

The MQPUT call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

Hobj (MQHOBJ) – input
Object handle.

This handle represents the queue to which the message is added. The value of Hobj was returned by a previous MQOPEN call that specified the MQOO_OUTPUT option.

MsgDesc (MQMD) – input/output
Message descriptor.

This structure describes the attributes of the message being sent, and receives information about the message after the put request is complete. See "MQMD – Message descriptor" on page 314 for details.

PutMsgOpts (MQPMO) – input/output
Message descriptor.

Options that control the action of MQPUT. See "MQPMO – Put message options" on page 319 for details.

BufferLength (MQLONG) – input
Length of the message in Buffer.

Zero is valid, and indicates that the message contains no application data.

Buffer (MQBYTE×BufferLength) – input
Message data.

This is a buffer containing the application data to be sent. The buffer should be aligned on a boundary appropriate to the nature of the data in the message. 4-byte alignment should be suitable for most messages (including messages containing MQ header structures), but some messages may require more stringent alignment.

If Buffer contains character and/or numeric data, the CodedCharSetId and Encoding fields in the MsgDesc parameter should be set to the values appropriate to the data; this will enable the receiver of the message to convert the data (if necessary) to the character set and encoding used by the receiver.

Note: All of the other parameters on the MQPUT call must be in the character set and encoding of the local queue manager (given by the CodedCharSetId queue-manager attribute and MQENC_NATIVE, respectively).

In the C programming language, the parameter is declared as a pointer-to-void; this means that the address of any type of data can be specified as the parameter. If the BufferLength parameter is zero, Buffer is not referred to; in this case, the parameter address passed by programs written in C can be null.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
`MQCC_OK`
Successful completion.
`MQCC_WARNING`
Warning (partial completion).
`MQCC_FAILED`
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
`MQRC_NONE`
(0, X'000') No reason to report.

If *CompCode* is MQCC_WARNING:
MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801') Message Priority exceeds maximum value supported.

If *CompCode* is MQCC_FAILED:
MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Buffer length parameter not valid.
MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_EXPIRY_ERROR
(2013, X'7DD') Expiry time not valid.
MQRC_FEEDBACK_ERROR
(2014, X'7DE') Feedback code not valid.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.
MQRC_HOBJ_ERROR
(2019, X'7E3') Object handle not valid.
MQRC_MD_ERROR
(2026, X'7EA') Message descriptor not valid.
MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Missing reply-to queue.
MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Message length greater than maximum for queue.
MQRC_MSG_TYPE_ERROR
(2029, X'7ED') Message type in message descriptor not valid.
MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') Queue not open for output.
MQRC_OPTIONS_ERROR
(2046, X'7FE') Options not valid or not consistent.
MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistence not valid.
MQRC_PMO_ERROR
(2173, X'87D') Put-message options structure not valid.
MQRC_PRIORITY_ERROR
(2050, X'802') Message priority not valid.
MQRC_PUT_INHIBITED
(2051, X'803') Put calls inhibited for the queue.
MQRC_Q_FULL
(2053, X'805') Queue already contains maximum number of messages.
MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808') No space available on disk for queue.
MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Report options in message descriptor not valid.
MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.
MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.
MQRC_UNKNOWN_CCSID
(2115, X'843') Unknown CCSID.

Language invocations

This call is supported in the following programming languages:
C invocation

```
MQPUT (Hconn,Hobj,&MsgDesc,&PutMsgOpts,BufferLength,Buffer,
&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQHOBJ  Hobj;           /*Object handle */
MQMD    MsgDesc;        /*Message descriptor */
MQPMO   PutMsgOpts;     /*Options that control the action of MQPUT */
```

```
MQLONG  BufferLength;   /*Length of the message in Buffer */
MQBYTE  Buffer [n ];    /*Message data */
MQLONG  CompCode;       /*Completion code */
MQLONG  Reason;         /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQPUT'USING HCONN,HOBJ,MSGDESC,PUTMSGOPTS,
BUFFERLENGTH,BUFFER,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object handle
  01 HOBJ PIC S9(9)BINARY.
**Message descriptor
  01 MSGDESC.
  COPY CMQMDV.
**Options that control the action of MQPUT
  01 PUTMSGOPTS.
  COPY CMQPMOV.
**Length of the message in Buffer
  01 BUFFERLENGTH PIC S9(9)BINARY.
**Message data
  01 BUFFER PIC X(n).
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQPUT (HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,BUFFER,
COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN        FIXED BIN(31); /*Connection handle */
DCL HOBJ         FIXED BIN(31); /*Object handle */
DCL MSGDESC      LIKE MQMD;     /*Message descriptor */
DCL PUTMSGOPTS   LIKE MQPMO;    /*Options that control the action of
                                    MQPUT */
DCL BUFFERLENGTH FIXED BIN(31); /*Length of the message in Buffer */
DCL BUFFER       CHAR(N);       /*Message data */
DCL COMPCODE     FIXED BIN(31); /*Completion code */
DCL REASON       FIXED BIN(31); /*Reason code qualifying CompCode */
```

MQPUT1 – Put one message

The MQPUT1 call puts one message on a queue. The queue need not be open.

Syntax

```
MQPUT1 (Hconn,ObjDesc,MsgDesc,PutMsgOpts,BufferLength,
Buffer,CompCode,Reason)
```

Parameters

The MQPUT1 call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

This handle represents the connection to the queue manager. The value of Hconn was returned by a previous MQCONN call.

ObjDesc (MQOD) – input/output
Object descriptor.

This is a structure which identifies the queue to which the message is added. See “MQOD – Object descriptor” on page 318 for details.

If security is enabled, the user must be authorized to open the queue for output.

MsgDesc (MQMD) – input/output
Message descriptor.

This structure describes the attributes of the message being sent, and receives feedback information after the put request is complete. See "MQMD – Message descriptor" on page 314 for details.

PutMsgOpts (MQPMO) – input/output
Message descriptor.

Options that control the action of MQPUT1. See "MQPMO – Put message options" on page 319 for details.

BufferLength (MQLONG) – input
Length of the message in Buffer.

Zero is valid, and indicates that the message contains no application data.

Buffer (MQBYTE×BufferLength) – input
Message data.

This is a buffer containing the application data to be sent. The buffer should be aligned on a boundary appropriate to the nature of the data in the message. 4-byte alignment should be suitable for most messages (including messages containing MQ header structures), but some messages may require more stringent alignment.

If Buffer contains character and/or numeric data, the CodedCharSetId and Encoding fields in the MsgDesc parameter should be set to the values appropriate to the data; this will enable the receiver of the message to convert the data (if necessary) to the character set and encoding used by the receiver.

Note: All of the other parameters on the MQPUT call must be in the character set and encoding of the local queue manager (given by the CodedCharSetId queue-manager attribute and MQENC_NATIVE, respectively).

In the C programming language, the parameter is declared as a pointer-to-void; this means that the address of any type of data can be specified as the parameter. If the BufferLength parameter is zero, Buffer is not referred to; in this case, the parameter address passed by programs written in C can be null.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
`MQCC_OK`
Successful completion.
`MQCC_WARNING`
Warning (partial completion).
`MQCC_FAILED`
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_WARNING:

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801') Message Priority exceeds maximum value supported.

If *CompCode* is MQCC_FAILED:
MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') Alias base queue not a valid type.
MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Buffer length parameter not valid.
MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_EXPIRY_ERROR
(2013, X'7DD') Expiry time not valid.
MQRC_FEEDBACK_ERROR
(2014, X'7DE') Feedback code not valid.
MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') No more handles available.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.
MQRC_MD_ERROR
(2026, X'7EA') Message descriptor not valid.
MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Missing reply-to queue.
MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Message length greater than maximum for queue.
MQRC_MSG_TYPE_ERROR
(2029, X'7ED') Message type in message descriptor not valid.
MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Object type not valid.
MQRC_OD_ERROR
(2044, X'7FC') Object descriptor structure not valid.
MQRC_OPTIONS_ERROR
(2046, X'7FE') Options not valid or not consistent.
MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistence not valid.
MQRC_PMO_ERROR
(2173, X'87D') Put-message options structure not valid.
MQRC_PRIORITY_ERROR
(2050, X'802') Message priority not valid.
MQRC_PUT_INHIBITED
(2051, X'803') Put calls inhibited for the queue.
MQRC_Q_FULL
(2053, X'805') Queue already contains maximum number of messages.
MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808') No space available on disk for queue.
MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Report options in message descriptor not valid.
MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.
MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.
MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822') Unknown alias base queue.
MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825') Unknown object name.
MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826') Unknown object queue manager.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827') Unknown remote queue manager.
MQRC_TARGET_CCSID_ERROR
(2115, X'843') Target coded character set identifier not valid.

Usage notes
Both the MQPUT and MQPUT1 calls can be used to put messages on a queue; which call to use depends on the circumstances:

- The MQPUT call should be used when multiple messages are to be placed on the same queue.

  An MQOPEN call specifying the MQOO_OUTPUT option is issued first, followed by one or more MQPUT requests to add messages to the queue; finally the queue is closed with an MQCLOSE call. This gives better performance than repeated use of the MQPUT1 call.
- The MQPUT1 call should be used when only one message is to be put on a queue. This call encapsulates the MQOPEN, MQPUT, and MQCLOSE calls into a single call, thereby minimizing the number of calls that must be issued.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQPUT1 (Hconn,&ObjDesc,&MsgDesc,&PutMsgOpts,
BufferLength,Buffer,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQOD ObjDesc;           /*Object descriptor */
MQMD MsgDesc;           /*Message descriptor */
MQPMO PutMsgOpts;       /*Options that control the action of MQPUT1 */
MQLONG BufferLength;    /*Length of the message in Buffer */
MQBYTE Buffer[n];       /*Message data */
MQLONG CompCode;        /*Completion code */
MQLONG Reason;          /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQPUT1'USING HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,
BUFFERLENGTH,BUFFER,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object descriptor
  01 OBJDESC.
  COPY CMQODV.
**Message descriptor
  01 MSGDESC.
  COPY CMQMDV.
**Options that control the action of MQPUT1
  01 PUTMSGOPTS.
  COPY CMQPMOV.
**Length of the message in Buffer
  01 BUFFERLENGTH PIC S9(9)BINARY.
**Message data
  01 BUFFER PIC X(n).
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PIC S9(9)BINARY.
```

PL/I invocation

```
CALL MQPUT1 (HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,BUFFER,
COMPCODE,REASON);
```

Declare the parameters as follows:

```
dcl Hconn        fixed bin(31);  /*Connection handle */
dcl ObjDesc      like MQOD;      /*Object descriptor */
dcl MsgDesc      like MQMD;      /*Message descriptor */
dcl PutMsgOpts   like MQPMO;     /*Options that control the action of
                                   MQPUT1 */
dcl BufferLength fixed bin(31);  /*Length of the message in Buffer */
dcl Buffer       char(n);        /*Message data */
dcl CompCode     fixed bin(31);  /*Completion code */
dcl Reason       fixed bin(31);  /*Reason code qualifying CompCode */
```

MQSET – Set object attributes

The MQSET call is used to change the attributes of an object represented by a handle. The object must be a queue.

Note: Once you have issued this call, if you issue a rollback, changes made to the MQSeries configuration will be reversed. However, the queue manager's internal control blocks will retain the changes created by the MQSET call. To remove the changes, reissue the MQSET call with the original values.

Syntax

```
MQSET (Hconn,Hobj,SelectorCount,Selectors,IntAttrCount,
IntAttrs,CharAttrLength,CharAttrs,CompCode,Reason)
```

Parameters

The MQSET call has the following parameters:

Hconn (MQHCONN) – input
Connection handle.

The value of *Hconn* was returned by a previous MQCONN call.

Hobj (MQHOBJ) – input
Object handle.

This handle represents the queue object whose attributes are to be set. The handle was returned by a previous MQOPEN call that specified the MQOO_SET option.

SelectorCount (MQLONG) – input
Count of selectors.

This is the count of selectors that are supplied in the Selectors array. It is the number of attributes that are to be set. Zero is a valid value. The maximum number allowed is 256.

Selectors (MQLONG×*SelectorCount*) – input
Array of attribute selectors.

This is an array of SelectorCount attribute selectors; each selector identifies an attribute (integer or character) whose value is to be set.

Each selector must be valid for the type of queue that Hobj represents. Only certain MQIA_* and MQCA_* values are allowed; these values are listed below.

Selectors for all types of queue

MQCA_Q_DESC
Queue description (MQ_Q_DESC_LENGTH).

MQIA_INHIBIT_PUT
Whether put operations are allowed. Can be one of the following values:

MQQA_PUT_ALLOWED
MQQA_PUT_INHIBITED

Selectors for local queues

MQCA_TRIGGER_CHANNEL_NAME
Channel name for MCA trigger process (MQ_CHANNEL_NAME_LENGTH).

MQCA_TRIGGER_DATA
Trigger user data (MQ_PROCESS_USER_DATA_LENGTH).

MQCA_TRIGGER_PROGRAM_NAME
Program name for trigger process (MQ_TRIGGER_PROGRAM_NAME_LENGTH).

MQCA_TRIGGER_TERM_ID
Terminal identifier for trigger process (MQ_TRIGGER_TERM_ID_LENGTH).

MQCA_TRIGGER_TRANS_ID
Transaction identifier for trigger process (MQ_TRIGGER_TRANS_ID_LENGTH).

MQIA_INHIBIT_GET
Whether get operations are allowed. Can be one of the following values:
MQQA_GET_ALLOWED
MQQA_GET_INHIBITED

MQIA_MAX_GLOBAL_LOCKS
Buffer size for queue manager to manage concurrent queue access.

MQIA_MAX_LOCAL_LOCKS
Buffer size for applications to manage concurrent queue access.

MQIA_MAX_MSG_LENGTH
Maximum message length for queue messages.

MQIA_MAX_Q_DEPTH
Maximum allowable queue depth for queues.

MQIA_MAX_Q_TRIGGERS
Maximum number of concurrent trigger instances for a particular queue.

MQIA_Q_USERS
Maximum number of concurrent opens per queue.

MQIA_SHAREABILITY
Queue shareability mode. Can be one of the following values:
MQQA_SHAREABLE
MQQA_NOT_SHAREABLE

MQIA_TRIGGER_CONTROL
Whether a trigger is required for the queue. Can be one of the following values:
MQTC_OFF
MQTC_ON

MQIA_TRIGGER_RESTART
Indicator for the reactivation of a trigger process. Can be one of the following values:
MQTRIGGER_RESTART_YES
MQTRIGGER_RESTART_NO

MQIA_TRIGGER_TYPE
Trigger event type. Can be one of the following values:
MQTT_NONE
MQTT_FIRST
MQTT_EVERY

MQIA_USAGE
Queue usage. Can be one of the following values:
MQUS_NORMAL
MQUS_TRANSMISSION
MQIA_Q_DEPTH_HIGH_LIMIT
High limit for queue depth.
MQIA_Q_DEPTH_LOW_LIMIT
Low limit for queue depth.
MQIA_Q_DEPTH_MAX_EVENT
Control attribute for queue depth max events.
MQIA_Q_DEPTH_HIGH_EVENT
Control attribute for queue depth high events.
MQIA_Q_DEPTH_LOW_EVENT
Control attribute for queue depth low events.
MQIA_Q_SERVICE_INTERVAL
Limit for queue service interval.
MQIA_Q_SERVICE_INTERVAL_EVENT
Control attribute for queue service interval events.

Selectors for remote queues
MQCA_REMOTE_Q_MGR_NAME
Name of remote queue manager (MQ_Q_MGR_NAME_LENGTH).
MQCA_REMOTE_Q_NAME
Name of remote queue as known on remote queue manager (MQ_Q_NAME_LENGTH).
MQCA_XMIT_Q_NAME
Name of local transmission queue (MQ_Q_NAME_LENGTH).

Selectors for alias queues
MQCA_BASE_Q_NAME
Name of queue that alias resolves to (MQ_Q_NAME_LENGTH).
MQIA_INHIBIT_GET
Whether get operations are allowed. Can be one of the following values:
MQQA_GET_ALLOWED
MQQA_GET_INHIBITED

IntAttrCount (MQLONG) – input
Count of integer attributes.

This is the number of elements in the IntAttrs array, and must be at least the number of MQIA_* selectors in the Selectors parameter. Zero is a valid value if there are none.

IntAttrs (MQLONG×IntAttrCount) – input
Array of integer attributes.

This is an array of IntAttrCount integer attribute values. These attribute values must be in the same order as the MQIA_* selectors in the Selectors array.

If the IntAttrCount or SelectorCount parameter is zero, IntAttrs is not referred to; in this case, the parameter address passed by programs written in C may be null.

CharAttrLength (MQLONG) – input
Length of character attributes buffer.

This is the length in bytes of the CharAttrs parameter and for MQ/VSE must zero.

CharAttrs (MQCHAR×CharAttrLength) – input
Character attributes.

This is not referred to by MQ/VSE. If programs are written in C then the parameter address passed by programs written in C may be null.

CompCode (MQLONG) – output
Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Length of character attributes not valid.
MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Character attributes string not valid.
MQRC_CICS_WAIT_FAILED
(2140, X'85C') Wait request rejected by CICS.
MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connection to queue manager lost.
MQRC_HCONN_ERROR
(2018, X'7E2') Connection handle not valid.
MQRC_HOBJ_ERROR
(2019, X'7E3') Object handle not valid.
MQRC_INHIBIT_VALUE_ERROR
(2020, X'7E4') Value for inhibit-get or inhibit-put queue attribute not valid.
MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Count of integer attributes not valid.
MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Integer attributes array not valid.
MQRC_NOT_OPEN_FOR_SET
(2040, X'7F8') Queue not open for set.
MQRC_OBJECT_CHANGED
(2041, X'7F9') Object definition changed since opened.
MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Queue manager name not valid or not known.
MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Queue manager not available for connection.
MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Count of selectors not valid.
MQRC_SELECTOR_ERROR
(2067, X'813') Attribute selector not valid.
MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Count of selectors too big.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Insufficient storage available.
MQRC_UNEXPECTED_ERROR
(2195, X'893') Unexpected error occurred.

Language invocations

This call is supported in the following programming languages:

C invocation

```
MQSET (Hconn,Hobj,SelectorCount,Selectors,IntAttrCount,IntAttrs,
CharAttrLength,CharAttrs,&CompCode,&Reason);
```

Declare the parameters as follows:

```
MQHCONN Hconn;          /*Connection handle */
MQHOBJ  Hobj;           /*Object handle */
MQLONG  SelectorCount;  /*Count of selectors */
MQLONG  Selectors[n];   /*Array of attribute selectors */
MQLONG  IntAttrCount;   /*Count of integer attributes */
MQLONG  IntAttrs[n];    /*Array of integer attributes */
MQLONG  CharAttrLength; /*Length of character attributes buffer */
MQCHAR  CharAttrs[n];   /*Character attributes */
MQLONG  CompCode;       /*Completion code */
MQLONG  Reason;         /*Reason code qualifying CompCode */
```

COBOL invocation

```
CALL 'MQSET'USING HCONN,HOBJ,SELECTORCOUNT, SELECTORS-
TABLE,INTATTRCOUNT,INTATTRS-TABLE,
CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON.
```

Declare the parameters as follows:

```
**Connection handle
  01 HCONN PIC S9(9)BINARY.
**Object handle
  01 HOBJ PIC S9(9)BINARY.
**Count of selectors
  01 SELECTORCOUNT PIC S9(9)BINARY.
**Array of attribute selectors
  01 SELECTORS-TABLE.
     02 SELECTORS PIC S9(9)BINARY OCCURS n TIMES.
**Count of integer attributes
  01 INTATTRCOUNT PIC S9(9)BINARY.
**Array of integer attributes
  01 INTATTRS-TABLE.
     02 INTATTRS PIC S9(9)BINARY OCCURS n TIMES.
**Length of character attributes buffer
  01 CHARATTRLENGTH PIC S9(9)BINARY.
**Character attributes
  01 CHARATTRS PIC X(n).
**Completion code
  01 COMPCODE PIC S9(9)BINARY.
**Reason code qualifying CompCode
  01 REASON PICS9(9)BINARY.
```

PL/I invocation

```
CALL MQSET (HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,
INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON);
```

Declare the parameters as follows:

```
DCL HCONN          FIXED BIN(31);  /*Connection handle */
DCL HOBJ           FIXED BIN(31);  /*Object handle */
DCL SELECTORCOUNT  FIXED BIN(31);  /*Count of selectors */
DCL SELECTORS(N)   FIXED BIN(31);  /*Array of attribute selectors */
DCL INTATTRCOUNT   FIXED BIN(31);  /*Count of integer attributes */
DCL INTATTRS(N)    FIXED BIN(31);  /*Array of integer attributes */
DCL CHARATTRLENGTH FIXED BIN(31);  /*Length of character attributes
                                             buffer */
```

```
DCL CHARATTRS      CHAR(N);        /*Character attributes */
DCL COMPCODE       FIXED BIN(31);  /*Completion code */
DCL REASON         FIXED BIN(31);  /*Reason code qualifying CompCode */
```

Attributes of MQSeries objects

In MQSeries for VSE/ESA, the attributes of all objects are as described in the *MQSeries Application Programming Reference* manual, with the following exception:

- Attributes of process definitions do not apply
- Attributes of Namelist definitions do not apply
- Attributes of AuthInfo definitions do not apply

The platform constant MQAT_CICS_VSE applies, value 10L.

Reason codes

In MQSeries for VSE/ESA, the reason codes (MQRC_) are described in older versions of the MQSeries Application Programming Reference manual or in Websphere MQ Messages (GC34-6057).

Appendix C. Application Programming Guidance

This appendix describes:
- Application program support
- Samples
- Syncpointing
- Triggers

Supporting application programs that use the MQI

MQSeries application programs need specific objects before they can run successfully. For example, Figure 62 shows an application that removes messages from a queue, processes them, and then sends some results to another queue on the same queue manager.

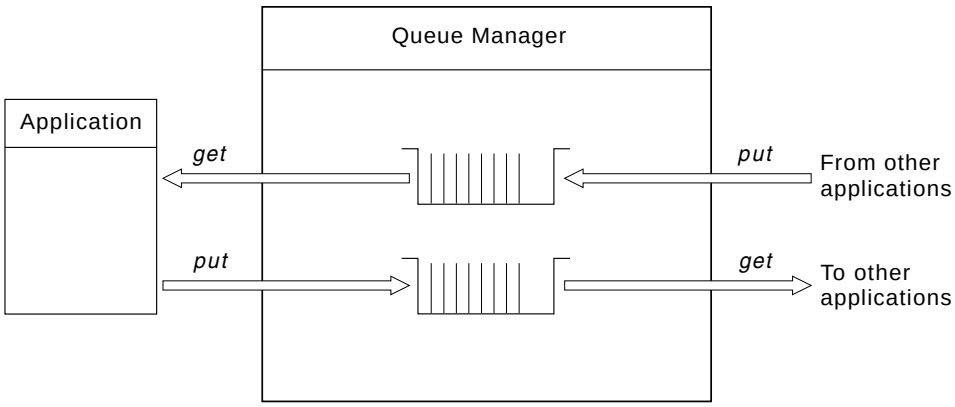

Figure 62. Queues, messages, and applications

Whereas applications can put (using MQPUT) messages on local or remote queues, they can only get (using MQGET) messages directly from local queues.

Before this application can run, these conditions must be satisfied:
- The queue manager must exist and be running.
- The first application queue, from which the messages are to be removed, must be defined.
- The second queue, on which the application puts the messages, must also be defined.
- The application must be able to connect to the queue manager. To do this the queue manager must be linked to the product code.
- The applications that put the messages on the first queue must also connect to a queue manager. If they are remote, they must also be set up with transmission queues and channels. This part of the system is not shown in Figure 62.

© Copyright IBM Corp. 1995, 2004

Sample source code provided

One COBOL-language sample trigger program, MQPECHO is provided with MQSeries for VSE/ESA. The source code for this program is shown in “Sample program MQPECHO.Z” on page 425, or it can be listed directly from the distribution files. Within the source code for MQPECHO, you can find examples that illustrate the use of the MQI calls in a trigger program.

In addition there are three sample programs, TTPTST1, TTPTST2, and TTPTST3. COBOL language copybook files are provided with the distribution file in the PRD2.MQSERIES library. These files provide examples of all of the MQI calls.

Compiling your application program

The MQI calls are provided in the library PRD2.MQSERIES.

Compilation

Ensure that you include the PRD2.MQSERIES library as part of the application phase step.

Developing applications in the C and PL/I programming languages

For CICS, COBOL is the language in which the MQSeries interface is written. Applications written in COBOL for VSE have been thoroughly tested with MQSeries. Sample programs and copybooks are provided in COBOL for VSE.

However, for a variety of reasons, you may need to write in another programming language. In these cases, you must meet the requirements of the COBOL language interface.

There are no sample programs provided in any other language, however, there are equivalents to the COBOL copybooks to enable applications to be built in other languages.

For the PL/I programming language, the following include files are provided:

CMQEPP.P	Declares the MQI calls and structures
CMQP.P	Declares the MQI constants
CMQCFP.P	Constants for MQI and PCF
CMQXP.P	Constants for MQI and PCF

For the C programming language, the following header files are provided:

CMQC.H	MQI header file.
CMQCFC.H	Constants for MQI and PCF.
CMQXC.H	Constants for MQI and PCF.

Application design guidelines

One of the key benefits provided by MQSeries is the ability for a distributed application to be developed that is totally independent of the underlying network. This network independence means that there is no need for an application to be aware of:

- The lower levels of the communication protocols, or
- The physical location of other applications on the network.

In order to take full advantage of this network independence, you must choose the queue names used by the application with care.

In particular, you are recommended to use a single logical name only, in your application programs, to refer to each MQSeries queue. For the MQSeries calls, this means only the Queue_Name field is used to identify queues. The use of the queue's fully qualified name (which includes both the Queue_Name field and the Queue_Manager_Name field) is not recommended.

The same is true when addressing MQSeries queues. As the Queue_Manager_Name is typically associated with a particular system, its use implies knowledge of the physical network.

Note: You are strongly recommended to use the Queue_Name field as the only logical queue name. This usage maximizes application flexibility and network independence. The mapping of the queue name in this form to the proper network destination then becomes a configuration issue to be handled by the MQSeries system administrator.

Syncpoints and triggers

This section describes syncpoints and triggers.

- Syncpoints allow an application to perform a series of changes, where the changes are treated as though they were a single change. They are described in "Syncpoint considerations."
- Triggers allow applications to be started automatically when messages arrive. They are described in "Triggers" on page 361.

Syncpoint considerations

Most applications need to access resources of one form or another, and a common requirement is to be able to make a coordinated set of changes to two or more resources.

"Coordinated" means that either all of the changes made to the resources take effect, or none of the changes takes effect. For some applications, queues need to be coordinated. Applications need to be able to get and put messages (and possibly update other resources, for example, databases), and know that either all of the operations take effect, or that none of the operations takes effect.

This set of coordinated operations is called a unit of work. An example of a unit of work is a debit and credit for a funds transfer in a financial application. Both operations must complete, or neither operation must complete, for a valid financial transaction to be completed.

Units of work: A unit of work starts when the first recoverable resource is affected. For message queuing, a unit of works starts when a message get or put occurs under syncpoint control.

The unit of work ends when either the application ends, or when the application declares a syncpoint.

If the unit of work is ended by an application ending, another unit of work can start. One instance of an application can be involved with several sequential units of work.

When a syncpoint is declared, any party (applications and the queue manager) that has interest in the unit of work can vote "yes" to commit the work, or "no", to back out of the unit of work.

Applications declare syncpoints, and register their votes, by issuing an environment-dependent call. It is advisable that an application should process CICS SYNCPOINT, followed by an MQCMIT call, prior to invoking an MQCLOSE call.

Participation of the MQGET, MQPUT, and MQPUT1 calls in the current unit of work is determined by the environment.

Distributed units of work (involving more than one queue manager) are not supported. A unit of work can contain queuing operations at only one instance of the queue manager.

If a message is put to a remote queue (that is, one on another queuing system), the action of the put request can be within the unit of work on the the system that puts the message but the arrival of the message on the target (remote) queue is outside its scope.

The get request for the message on the remote queue can be within the scope of work on that system, but the two units of work are not related by the queue manager.

Putting messages within a unit of work: If an MQPUT or MQPUT1 call participates in the current unit of work, the message is not available for retrieval from the target queue, between the completion of the MQPUT call and the successful completion of the unit of work. The only exception to this rule is if the target queue is within the same unit of work as the one within which it was put.

Only when, and if, the unit of work is committed successfully does the message become generally available.

Any errors detected by the queue manager when the message is put are returned to the application immediately, by means of the completion code and reason code parameters. Errors that can be detected in this way include:
- Message too large for queue
- Queue full
- Put requests inhibited for queue

Failure to put the message does not affect the status of the unit of work, because that message is not part of the unit of work. The application can still commit or backout of the unit of work as required.

However, should an application fail after a message was put successfully within a unit of work, the transaction is backed out.

Getting messages within a unit of work: If an MQGET call participates in the current unit of work, between the completion of the MQGET call and the successful completion of the unit of work, the message remains on the queue but becomes invisible.

Neither the application that retrieved the message, nor any other application serving the queue, can see or obtain the message again. If the unit of work is committed successfully, the message is deleted from the queue. However, if the unit of work is backed out, the message is reinstated in the queue in its original position, and becomes available to the same or another application to retrieve.

Syncpoint and persistence: Only persistent messaging is supported. Persistent messages do not get deleted if the queue manager is restarted. Therefore, they are fully recovered when the queue manager is restarted. Syncpointing by the application causes these records to be in a logical unit of work. Any records that were syncpointed are still recovered if the queue manager is shutdown and restarted.

Syncpoint Rollback

If your application wants to undo what has been done since the beginning of the current logical unit of work, it has to issue the following command:

```
EXEC CICS SYNCPOINT ROLLBACK
```

This can have the following, non-desired, results:

- Monitoring shows incorrect queue depth values, because the queue manager is not aware of rollbacks. This value is correctly reset when stopping and restarting the queue manager.
- The queue depth and the last sequence number are not the same anymore. Even if a message has been rolled back, its message sequence number (MSN) is never used again. This is because other applications may have also put messages into the same queue. For example:

```
Transaction A writes Message number  5
Transaction B    .................   6
Transaction A    .................   7
Transaction C    .................   8
```

  At this point the queue depth is 8. Assume Transaction A rolls back, in which case messages 5 and 7 will be never retrieved. Note that this is not an error. The queue depth is now 6, and the next MSN will be 9.

  From an application point of view this has no impact at all, but can be surprising when using the MQMT dialogs.

  Note: To be able to use SYNCPOINT ROLLBACK, you MUST use a CICS System LOG file, that is, define a CICS JCT.

Triggers

Some applications run continuously, and are always available to read a message when it arrives on the application's input queue. However, keeping the application active consumes system resources, even when the application is waiting for a message to arrive. This additional load on the system is not desirable. Instead of the application running continuously, the application is designed to run only when there are messages to be processed. The queue manager's triggering facility is used to help make this happen.

Overview of triggering

A local queue definition can have a trigger event associated with it when it is defined. This event is defined to activate the MQ trigger API Handler, that is, the MQ02 CICS Transaction.

The trigger API handler does either a CICS LINK to the application program or a CICS START to the application transaction. This is based on whether you defined a program name or a transaction name in the queue definition.

When an application program is entered, an information area is available. This area can be mapped by using the structure defined in the member CMQTMV.C:

1. If the trigger facility specified a program name, this area is passed using the COMMAREA.

   To return to the API handler, you should issue an EXEC CICS RETURN.
2. If the trigger facility specified a transaction name, this information area can be accessed by issuing an EXEC CICS RETRIEVE command.

   Before exiting from the program, you must issue an MQCLOSE command.

Note: In order to perform this function, this transaction ID must be unique in respect to any MQSeries system local queue. Essentially, the MQSeries system queue manager recognizes this transaction ID as a local queue being opened. When this queue is closed fully, this trigger event will be closed, allowing another trigger for this queue to be activated.

Trigger conditions

The queue manager activates a trigger event based on the event type defined for the current queue, against which the MQPUT operation has been requested.

Note: If a non-empty queue is stopped and restarted, the trigger condition suffices, regardless of the trigger event type.

The trigger API handler waits until this MQPUT request has been completed. This implies that the MQPUT request can be successful or unsuccessful, that is, rolled back. The activated trigger application program should perform an MQGET call.

If the result of this MQGET call is an empty condition, that is, MQRC_NO_MSG_AVAILABLE, the original application current logical unit-of-work has been rolled back. It is up to the application trigger program to determine whether to continue to wait or just end.

A trigger event type of “FIRST” generates a trigger event after the queue goes from an empty status to a non-empty one. Therefore, any application triggered in this manner must process the queue until the queue is empty.

A trigger event type of “EVERY” generates a trigger event after every MQPUT call has been completed, up to the maximum number of trigger events specified on the Extended Local Queue Configuration screen. See “Local queue extended definition screen” on page 79 for further information.

Defining a sender channel component

A sender channel component causes the channel to start if there are messages on the transmission queue to be sent to the remote node.

In contrast, a server channel component will not start unless started by a remote requester component, or by manual intervention, even when there are messages to be sent.

On the transmission queue for the sender channel, code the fields as follows:
- Usage Mode – T
- Trigger Enable – Y
- Trigger Type – E
- Max Trigger Starts – 1
- Transaction ID – <blanks>
- Program ID – MQPSEND
- Remote CID – <the name of the channel>

Note: MQSeries for VSE/ESA does not support requester channels.

Defining a program to be triggered

This technique is used when an application program is to receive messages from the MQSeries system queue manager in the manner described in "Overview of triggering" on page 361 for a CICS LINK.

- Usage Mode – N
- Trigger Enable – Y
- Trigger Type – E or F
- Max Trigger Starts – 1
- Transaction ID – <blanks>
- Program ID – <application program name>
- Remote CID – <blanks>
- User data – <optional data for trigger program>

Defining a transaction to be triggered

"Overview of triggering" on page 361, for CICS START, provides details of how to trigger a program based on its transaction ID. Note, that the transaction should not be invoked outside the trigger mechanism. However, by defining a different transaction name with the same program name, the program can be invoked outside the trigger environment.

Code as follows in the queue definition:

- Usage Mode – N
- Trigger Enable – Y
- Trigger Type – E or F
- Max Trigger Starts – 1
- Transaction ID – <user Transaction>
- Program ID – <blanks>
- Remote CID – <blanks>
- User data – <optional data for trigger program>

Appendix D. Sample JCL and programs

This appendix lists sample JCL and COBOL-language programs that are supplied with MQSeries for VSE/ESA V2.1.2.

Sample JCL to process MQPUTIL

```
* ** JOB JNM=MQJUTILY,DISP=D,CLASS=A
* ** LST DISP=H,CLASS=Q,PRI=3
// JOB MQJUTILY - Execute VSE/ESA MQ/Series Batch Utility Program.
* ------------------------------------------------------------------*
*      I M P O R T A N T    I M P O R T A N T  I M P O R T A N T    *
*                                                                   *
*   Please change :                                                 *
*                 "*  ** JOB" to  "* $$ JOB"                        *
*                 "*  ** LST" to  "* $$ LST"                        *
*                 "*  ** EOJ" to  "* $$ EOJ"                        *
*                                                                   *
* ------------------------------------------------------------------*
*   This job executes MQPUTIL to access the CONFIGURATION file      *
*                                                                   *
*   This file is a sample and needs modification to suit the        *
*   users environment.                                              *
*                                                                   *
* ------------------------------------------------------------------*
* Licensed Materials - Property of IBM                              *
* 5686-A06                                                          *
* (C) Copyright IBM Corp. 1998, 2002                                *
*                                                                   *
* US Government Users Restricted Rights - Use, duplication or       *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. *
* ------------------------------------------------------------------*
// DLBL CONFIG,'MQSERIES.MQFCNFG',,VSAM,CAT=MQMCAT
// DLBL INLOG,'MQSERIES.MQFLOG',,VSAM,CAT=MQMCAT
// LIBDEF PHASE,SEARCH=(PRD2.MQSERIES,PRD2.SCEEBASE)
// ASSGN SYS004,SYSIPT
// ASSGN SYS005,SYSLST
// EXEC MQPUTIL,SIZE=AUTO
*RESET MSN          00000002
*RESET CHECKPOINT   00000002
*PRINT CONFIG
*PRINT LOG
/*
/&
* ** EOJ
```

Sample program TTPTST2.Z

This program is a test facility for sending and receiving messages. It must be invoked by terminal input format as:

```
TST2 func nn queue-name
```

where:

TST2 Is the transaction ID.

func Is any of the following functions:
- BOTH Put and get messages.
- GET Get messages.
- GETD Get and delete messages.

© Copyright IBM Corp. 1995, 2004

INQ Invoke MQINQ to inquire about the attributes of queues.
PUT Put messages.
PUTR Put messages and send reply.
PUT1 Put and delete messages.

nn Is the number of messages to be processed, from 01 through 99.

queue name
Is the name of the local or transmission queue to be processed.

For example, `TST2 PUT 99 QUE1` puts 99 messages into a local queue named QUE1. All the messages read `THIS IS A MESSAGE TEXT`. Typing `TST2` without parameters causes help instructions to be displayed.

TTPTST2.Z

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    TTPTST2.
 AUTHOR.        IBM.

 DATE-COMPILED.
*-----------------------------------------------------------------*
*                                                                 *
*   Program:      TTPTST2                                         *
*   Description:  Sample program to illustrate the use of the     *
*                 MQ Message Queue Interface (MQI).               *
*                                                                 *
*-----------------------------------------------------------------*
/
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 DATA DIVISION.

 WORKING-STORAGE SECTION.

*-----------------------------------------------------------*

 01  WS-WORK-FIELDS.
     05  WS-IDX                   PIC S9(4)  COMP VALUE ZERO.
     05  WS-COUNT                 PIC S9(4)  COMP VALUE ZERO.
     05  WS-PROCESS-TIMES         PIC  9(4)       VALUE ZERO.
     05  WS-DURATION-SECS         PIC  X(8)       VALUE SPACES.
     05  WS-PASS-MSG-LENGTH       PIC S9(4)  COMP   VALUE ZERO.
     05  WS-APPL-MSG-LENGTH       PIC S9(8)  COMP   VALUE ZERO.
     05  WS-ABSTIME               PIC S9(15) COMP-3 VALUE ZERO.
     05  WS-ABSTIME2              PIC S9(15) COMP-3 VALUE ZERO.
     05  WS-DATE.
        10  WS-DATE-CC            PIC  99  VALUE ZERO.
        10  WS-DATE-YYMMDD.
          12  WS-DATE-YY          PIC  99  VALUE ZERO.
          12  WS-DATE-MM          PIC  99  VALUE ZERO.
          12  WS-DATE-DD          PIC  99  VALUE ZERO.
     05  WS-TIME-9                PIC  9(7)  VALUE ZERO.
     05  WS-TIME  REDEFINES  WS-TIME-9.
        10  FILLER                PIC  9.
        10  WS-TIME-HHMMSS.
          12  WS-TIME-HH          PIC  99.
          12  WS-TIME-MM          PIC  99.
          12  WS-TIME-SS          PIC  99.
     05  WS-FORMATTED-TIME.
         10  WS-FORMAT-TIME-HH    PIC  X(02) VALUE SPACES.
         10  FILLER               PIC  X(01) VALUE ':'.
         10  WS-FORMAT-TIME-MM    PIC  X(02) VALUE SPACES.
         10  FILLER               PIC  X(01) VALUE ':'.
         10  WS-FORMAT-TIME-SS    PIC  X(02) VALUE SPACES.
     05  WS-FORMATTED-DATE.
```

```
          10  WS-FORMAT-DATE-MM     PIC  X(02) VALUE SPACES.
          10  FILLER                PIC  X(01) VALUE '/'.
          10  WS-FORMAT-DATE-DD     PIC  X(02) VALUE SPACES.
          10  FILLER                PIC  X(01) VALUE '/'.
          10  WS-FORMAT-DATE-YY     PIC  X(02) VALUE SPACES.

      05  WS-QM-Q-NAME.
        10  WS-QM-NAME              PIC  X(48) VALUE 'QM1   '.
        10  WS-Q-NAME               PIC  X(48) VALUE 'QUEUE'.

      05  WS-REPLY-Q                PIC  X(48) VALUE 'QUE1'.

      05  WS-ERR-MSG-FLAG           PIC  X     VALUE SPACES.
       88 WS-ERR-MSG                           VALUE 'Y'.

      05  WS-STARTED-FLAG           PIC  X     VALUE SPACES.
       88 WS-STARTED                           VALUE 'Y'.

      05  WS-TIMESTAMP              PIC  X     VALUE SPACES.
       88 WS-PUT-TIMESTAMP                     VALUE 'Y'.

      05  WS-TIMESTAMP-VALUE.
         10  WS-TIMESTAMP-DATE      PIC X(6) VALUE SPACES.
         10  WS-TIMESTAMP-TIME      PIC X(6) VALUE SPACES.
         10  WS-TIMESTAMP-COUNT     PIC 999   VALUE ZERO.

      05  WS-STARTCODE               PIC  XX   VALUE SPACE.
        88  START-WITH-DATA         VALUE 'SD'.
        88  START-WITH-NO-DATA      VALUE 'S '.

      05  WS-END-OF-MESSAGES-FLAG   PIC  X  VALUE SPACES.
        88 WS-END-OF-MESSAGES              VALUE 'Y'.

      05  WS-TRUNCATED-MESSAGES-F   PIC  X  VALUE SPACES.
        88 WS-TRUNCATED-MESSAGES           VALUE 'Y'.

  *-------------------------------------------------------------*

   77  WS-DATA-FLENGTH              PIC S9(8) COMP VALUE ZERO.
   77  WS-DATA-LENGTH               PIC S9(4) COMP VALUE ZERO.
   01  WS-DATA-ALL.
       05  WS-DATA-WITH-QUEUE.
         10  WS-DATA-WITH-TIMES.
            12  WS-DATA-WITH-FUNCTION.
              15  WS-TRANSID         PIC  X(5)  VALUE 'TST2 '.
              15  WS-DATA-FUNCTION   PIC  XXXX.
              88 VALID-DATA-FUNCTION           VALUE 'PUT'
                                                     'GET'
                                                     'BOTH'
                                                     'PUT1'
                                                     'INQ'
                                                     'PUTR'
                                                     'GETD'
                                                     'HELP'
                                                     '?'.
              88 WS-HELP-FUN                   VALUE 'HELP' '?'.
              88 WS-PUT                        VALUE 'PUT'.
              88 WS-GET                        VALUE 'GET'.
              88 WS-BOTH                       VALUE 'BOTH'.
              88 WS-PUT1                       VALUE 'PUT1'.
              88 WS-PUT-WITH-REPLY             VALUE 'PUTR'.
              88 WS-GET-WITH-DELETE            VALUE 'GETD'.
              88 WS-INQ                        VALUE 'INQ'.

            12  FILLER               PIC  X     VALUE ' '.
            12  WS-DATA-TIMES        PIC  99    VALUE 01.
         10  WS-DATA-SYNC-FLAG       PIC  X     VALUE ' '.
```

```
      10  WS-DATA-QUEUE            PIC  X(48) VALUE SPACES.

 01  MQI-SELECTOR-COUNT.
    05 WS-SELECTOR-COUNT           PIC S9(8) COMP VALUE ZERO.

 01  MQI-SELECTOR.
    05  MQI-SELECTOR-ENTRY   OCCURS 40 TIMES


 01  MQI-IN-ATTR-COUNT.       01  MQI-IN-ATTR-COUNT.
    05 WS-IN-ATTR-COUNT         PIC S9(8) COMP VALUE +40.

 01  MQI-IN-ATTR.
    05 MQI-IN-ATTR-ENTRY   OCCURS 40 TIMES
                              PIC S9(8) COMP.

*-----------------------------------------------------------------*
 01  WS-PASSED-INFO.

     COPY   TTITST2.

  EJECT
*-----------------------------------------------------------------*
 01  WS-NEED-REPLY.
     05  FILLER                   PIC  X(80) VALUE
     'Please enter REPLY QUEUE name with trailing blanks or ErsEOF
-    ' (eg. Ctrl - Del)'.
  EJECT
*-----------------------------------------------------------------*
 01  WS-HELP.
     05  FILLER                   PIC  X(80) VALUE
     ' TST2 is a test facility for SENDING / RECEIVING messages'.
     05  FILLER                   PIC  X(80) VALUE
     ' The format of command is as follows:'.
     05  FILLER                   PIC  X(80) VALUE
     ' TST2 XXXX [NN] QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
-    'QQ'.
     05  FILLER                   PIC  X(80) VALUE SPACES.
     05  FILLER                   PIC  X(80) VALUE
     '(NOTE a single space or comma separates the params)'.
     05  FILLER                   PIC  X(80) VALUE
     '  XXXX  4-character function code, pad with trailing blank'.
     05  FILLER                   PIC  X(80) VALUE
     '                       HELP - DISPLAY THIS HELP TEXT'.
     05  FILLER                   PIC  X(80) VALUE
     '                       PUT  - MQPUT  MESSAGES'.
     05  FILLER                   PIC  X(80) VALUE
     '                       PUT1 - MQPUT1 MESSAGES'.
     05  FILLER                   PIC  X(80) VALUE
     '                       PUTR - MQPUT W/ REPLY MESSAGE'.
     05  FILLER                   PIC  X(80) VALUE
     '                       GET  - MQGET  MESSAGES'.
     05  FILLER                   PIC  X(80) VALUE
     '                       GETD - MQGET W/ BROWSE & DELETE'.
     05  FILLER                   PIC  X(80) VALUE
     '                       BOTH - MQPUT FOLLOWED BY MQGET'.
     05  FILLER                   PIC X(80)  VALUE
     '                       INQ  - INQ ABOUT QUEUE (no count NN)'.
     05  FILLER                   PIC  X(80) VALUE
     '  NN    2-digit number with leading zero (01 TO 99)'.
     05  FILLER                   PIC  X(80) VALUE
     '  QQQQ  A 48-character field giving the name of a queue.'.
     05  FILLER                   PIC  X(80) VALUE
     '  An additional prompt will ask for the name of the reply qu
-    'eue for PUTR option.'.
 01  WS-HELP-RED  REDEFINES   WS-HELP.
     05  WS-HELP-LINE  OCCURS 17 TIMES
```

```
                                PIC  X(80).
*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 01  WS-ALL-MSG.
     05  WS-OK-MSG.
         10  FILLER                  PIC  X(80) VALUE
          ' FULL CYCLE HAS BEEN PERFORMED SUCCESSFULLY'.
         10 WS-OK-MSG-1              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-2              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-3              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-4              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-5              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-6              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-7              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-8              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-9              PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-10             PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-11             PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-12             PIC  X(80) VALUE SPACES.
     05  WS-ERR-LINES.
         10  FILLER                  PIC  X(400) VALUE SPACES.

 01  WS-OK-STATS-LINE-1.
     05  FILLER                  PIC  X(20) VALUE
        '    QUEUE USED -'.
     05  WS-OK-QUEUE             PIC  X(48).

 01  WS-OK-STATS-LINE-2.
     05  FILLER                  PIC  X(20) VALUE
        ' REPLY Q-'.
     05  WS-OK-QUEUE-REPLY       PIC  X(48).

 01  WS-OK-STATS-LINE-3.
     05  FILLER                  PIC  X(40) VALUE
        '    NUMBER OF MESSAGES PROCESSED -'.
     05  WS-OK-MESSAGES          PIC Z99.

 01  WS-OK-STATS-LINE-4.
     05  FILLER                  PIC  X(40) VALUE
        '    TOTAL SECONDS ..... -'.
     05  WS-OK-TIME              PIC  X(8).
 01  WS-INQ-DETS.
     05 FILLER                   PIC X(6)
        VALUE 'TYPE: '.
     05 WS-I-TYPE                PIC 99.
     05 FILLER                   PIC X(10)
        VALUE ' INHIBIT: '.
     05 WS-I-INHIBIT             PIC 99.
     05 FILLER                   PIC X(7)
       VALUE ' MAXL: '.
     05 WS-I-MAXL                PIC 999999.

*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 01  WS-ERROR-MESSAGES.
     05 WS-ERR-DATA.
       10  FILLER                  PIC  X(13) VALUE
       '  DATA ERROR:'.
       10  FILLER                  PIC  X(9) VALUE
       '  LENGTH='.
       10  WS-ERR-DATA-LENGTH      PIC  9(8) VALUE ZERO.
       10  FILLER                  PIC  X(9) VALUE
       ', DATA  ='.
       10  WS-ERR-DATA-AREA        PIC  X(200) VALUE SPACES.
       10  FILLER                  PIC  X(4) VALUE
```

```
       '****'.

     05 WS-ERR-DISPLAY.
       10  FILLER                  PIC  X(13) VALUE
       '  MQ   ERROR:'.
       10  FILLER                  PIC  X(9) VALUE
       '  LEVEL ='.
       10  WS-LEVEL                PIC  X(8) VALUE SPACES.
       10  FILLER                  PIC  X(9) VALUE
       ', FUNC  ='.
       10  WS-FUNCTION             PIC  X(8) VALUE SPACES.
       10  FILLER                  PIC  X(9) VALUE
       ', CC    ='.
       10  WS-ERR-DISPLAY-CCODE    PIC  9(4) VALUE ZERO.
       10  FILLER                  PIC  X(9) VALUE
       ', RC    ='.
       10  WS-ERR-DISPLAY-RCODE    PIC  9(4) VALUE ZERO.
       10  FILLER                  PIC  X(4) VALUE
       '****'.
  EJECT
*-----------------------------------------------------------------*
 01  FILLER.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQV                                     **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQI constants        **
**                                                             **
**  VERSION 2.1.0                                              **
**                                                             **
**  FUNCTION:         This file declares the constants         **
**                    which form part of the IBM Message       **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************

*****************************************************************
**  Values Related to MQDLH Structure                          **
*****************************************************************
**   Structure Identifier
  10 MQDLH-STRUC-ID PIC X(4) VALUE 'DLH '.

**   Structure Version Number
  10 MQDLH-VERSION-1 PIC S9(9) BINARY  VALUE 1.


*****************************************************************
**  Values Related to MQGMO Structure                          **
*****************************************************************

**   Structure Identifier
  10 MQGMO-STRUC-ID PIC X(4) VALUE 'GMO '.

**   Structure Version Number
  10 MQGMO-VERSION-1 PIC S9(9) BINARY  VALUE 1.

**   Get-Message Options
  10 MQGMO-WAIT                    PIC S9(9) BINARY  VALUE 1.
  10 MQGMO-NO-WAIT                 PIC S9(9) BINARY  VALUE 0.
  10 MQGMO-BROWSE-FIRST            PIC S9(9) BINARY  VALUE 16.
  10 MQGMO-BROWSE-NEXT             PIC S9(9) BINARY  VALUE 32.
  10 MQGMO-ACCEPT-TRUNCATED-MSG    PIC S9(9) BINARY  VALUE 64.
  10 MQGMO-SET-SIGNAL              PIC S9(9) BINARY  VALUE 8.
  10 MQGMO-SYNCPOINT               PIC S9(9) BINARY  VALUE 2.
  10 MQGMO-NO-SYNCPOINT            PIC S9(9) BINARY  VALUE 4.
  10 MQGMO-MSG-UNDER-CURSOR        PIC S9(9) BINARY  VALUE 256.
  10 MQGMO-LOCK                    PIC S9(9) BINARY  VALUE 512.
```

```
  10 MQGMO-UNLOCK                  PIC S9(9) BINARY  VALUE 1024.
  10 MQGMO-CONVERT                 PIC S9(9) BINARY  VALUE 16384.

**   Wait Interval
  10 MQWI-UNLIMITED PIC S9(9) BINARY  VALUE -1.


****************************************************************
**  Values Related to MQMD Structure                          **
****************************************************************

**   Structure Identifier
  10 MQMD-STRUC-ID PIC X(4) VALUE 'MD  '.

**   Structure Version Number
  10 MQMD-VERSION-1 PIC S9(9) BINARY  VALUE 1.

**   Report Options
  10 MQRO-NONE PIC S9(9) BINARY  VALUE 0.

**   Message Types
  10 MQMT-REQUEST  PIC S9(9) BINARY  VALUE 1.
  10 MQMT-REPLY    PIC S9(9) BINARY  VALUE 2.
  10 MQMT-DATAGRAM PIC S9(9) BINARY  VALUE 8.
  10 MQMT-REPORT   PIC S9(9) BINARY  VALUE 4.

**   Expiry Value
  10 MQEI-UNLIMITED PIC S9(9) BINARY  VALUE -1.

**   Feedback Values
  10 MQFB-NONE         PIC S9(9) BINARY  VALUE 0.
  10 MQFB-QUIT         PIC S9(9) BINARY  VALUE 256.
  10 MQFB-SYSTEM-FIRST PIC S9(9) BINARY  VALUE 1.
  10 MQFB-SYSTEM-LAST  PIC S9(9) BINARY  VALUE 65535.
  10 MQFB-APPL-FIRST   PIC S9(9) BINARY  VALUE 65536.
  10 MQFB-APPL-LAST    PIC S9(9) BINARY  VALUE 999999999.

* format
  10 MQFMT-NONE                  PIC  X(8) VALUE SPACES.
  10 MQFMT-DEAD-LETTER-Q-HEADER  PIC  X(8) VALUE 'MQDLQH'.
  10 MQFMT-TRIGGER               PIC  X(8) VALUE 'MQTRIG'.
  10 MQFMT-XMIT-Q-HEADER         PIC  X(8) VALUE 'MQXMIT'.
  10 MQFMT-STRING                PIC  X(8) VALUE 'MQSTR'.
  10 MQFMT-ADMIN                 PIC  X(8) VALUE 'MQADMIN '.
  10 MQFMT-CICS                  PIC  X(8) VALUE 'MQCICS  '.
  10 MQFMT-COMMAND-1             PIC  X(8) VALUE 'MQCMD1  '.
  10 MQFMT-COMMAND-2             PIC  X(8) VALUE 'MQCMD2  '.
  10 MQFMT-DIST-HEADER           PIC  X(8) VALUE 'MQHDIST '.
  10 MQFMT-DEAD-LETTER-HEADER    PIC  X(8) VALUE 'MQDEAD  '.
  10 MQFMT-EVENT                 PIC  X(8) VALUE 'MQEVENT '.
  10 MQFMT-IMS                   PIC  X(8) VALUE 'MQIMS   '.
  10 MQFMT-IMS-VAR-STRING        PIC  X(8) VALUE 'MQIMSVS '.
  10 MQFMT-PCF                   PIC  X(8) VALUE 'MQPCF   '.
  10 MQFMT-REF-MSG-HEADER        PIC  X(8) VALUE 'MQHREF  '
  10 MQFMT-RF-HEADER             PIC  X(8) VALUE 'MQHRF   '
  10 MQFMT-SAP                   PIC  X(8) VALUE 'MQHSAP  '
  10 MQFMT-WORK-INFO-HEADER      PIC  X(8) VALUE 'MQHWIH  '

**   Encoding Value
  10 MQENC-NATIVE PIC S9(9) BINARY  VALUE 785.

**   Encoding Masks
  10 MQENC-INTEGER-MASK  PIC S9(9) BINARY  VALUE 15.
  10 MQENC-DECIMAL-MASK  PIC S9(9) BINARY  VALUE 240.
  10 MQENC-FLOAT-MASK    PIC S9(9) BINARY  VALUE 3840.
  10 MQENC-RESERVED-MASK PIC S9(9) BINARY  VALUE -4096.
```

```
**   Encodings for Binary Integers
  10 MQENC-INTEGER-UNDEFINED PIC S9(9) BINARY  VALUE 0.
  10 MQENC-INTEGER-NORMAL    PIC S9(9) BINARY  VALUE 1.
  10 MQENC-INTEGER-REVERSED  PIC S9(9) BINARY  VALUE 2.

**   Encodings for Packed-Decimal Integers
  10 MQENC-DECIMAL-UNDEFINED PIC S9(9) BINARY  VALUE 0.
  10 MQENC-DECIMAL-NORMAL    PIC S9(9) BINARY  VALUE 16.
  10 MQENC-DECIMAL-REVERSED  PIC S9(9) BINARY  VALUE 32.

**   Encodings for Floating-Point Numbers
  10 MQENC-FLOAT-UNDEFINED     PIC S9(9) BINARY  VALUE 0.
  10 MQENC-FLOAT-IEEE-NORMAL   PIC S9(9) BINARY  VALUE 256.
  10 MQENC-FLOAT-IEEE-REVERSED PIC S9(9) BINARY  VALUE 512.
  10 MQENC-FLOAT-S390          PIC S9(9) BINARY  VALUE 768.

**   Coded Character-Set Identifier
  10 MQCCSI-Q-MGR PIC S9(9) BINARY  VALUE 0.

**   Persistence Values
  10 MQPER-PERSISTENT           PIC S9(9) BINARY  VALUE 1.
  10 MQPER-PERSISTENCE-AS-Q-DEF PIC S9(9) BINARY  VALUE 2.

**   Message Id Value
  10 MQMI-NONE PIC X(24) VALUE LOW-VALUES.

**   Correllation Id Value
  10 MQCI-NONE PIC X(24) VALUE LOW-VALUES.


**********************************************************************
**  Values Related to MQOD Structure                                **
**********************************************************************

**   Structure Identifier
  10 MQOD-STRUC-ID PIC X(4) VALUE 'OD  '.

**   Structure Version Number
  10 MQOD-VERSION-1 PIC S9(9) BINARY  VALUE 1.

**   Object Types
  10 MQOT-Q PIC S9(9) BINARY  VALUE 1.


**********************************************************************
**  Values Related to MQPMO Structure                               **
**********************************************************************

**   Structure Identifier
  10 MQPMO-STRUC-ID PIC X(4) VALUE 'PMO '.

**   Structure Version Number
  10 MQPMO-VERSION-1 PIC S9(9) BINARY  VALUE 1.

**   Put-Message Options
  10 MQPMO-SYNCPOINT            PIC S9(9) BINARY  VALUE 2.
  10 MQPMO-NO-SYNCPOINT         PIC S9(9) BINARY  VALUE 4.

**********************************************************************
**  Values Related to MQTM Structure                                **
**********************************************************************

**   Structure Identifier
  10 MQTM-STRUC-ID PIC X(4) VALUE 'TM  '.

**   Structure Version Number
  10 MQTM-VERSION-1 PIC S9(9) BINARY  VALUE 1.
```

```
*****************************************************************
**  Values Related to MQCLOSE Call                             **
*****************************************************************

**   Close Options
  10 MQCO-NONE PIC S9(9) BINARY  VALUE 0.


*****************************************************************
**  Values Related to MQINQ Call                               **
*****************************************************************

**   Character-Attribute Selectors
  10 MQCA-BASE-Q-NAME       PIC S9(9) BINARY  VALUE 2002.
  10 MQCA-CREATION-DATE     PIC S9(9) BINARY  VALUE 2004.
  10 MQCA-CREATION-TIME     PIC S9(9) BINARY  VALUE 2005.
  10 MQCA-FIRST             PIC S9(9) BINARY  VALUE 2001.
  10 MQCA-INITIATION-Q-NAME PIC S9(9) BINARY  VALUE 2008.
  10 MQCA-LAST              PIC S9(9) BINARY  VALUE 4000.
  10 MQCA-PROCESS-NAME      PIC S9(9) BINARY  VALUE 2012.
  10 MQCA-Q-DESC            PIC S9(9) BINARY  VALUE 2013.
  10 MQCA-Q-NAME            PIC S9(9) BINARY  VALUE 2016.
  10 MQCA-REMOTE-Q-MGR-NAME PIC S9(9) BINARY  VALUE 2017.
  10 MQCA-REMOTE-Q-NAME     PIC S9(9) BINARY  VALUE 2018.

**   Integer-Attribute Selectors
  10 MQIA-CURRENT-Q-DEPTH   PIC S9(9) BINARY  VALUE 3.
  10 MQIA-DEF-PERSISTENCE   PIC S9(9) BINARY  VALUE 5.
  10 MQIA-DEFINITION-TYPE   PIC S9(9) BINARY  VALUE 7.
  10 MQIA-FIRST             PIC S9(9) BINARY  VALUE 1.
  10 MQIA-INHIBIT-GET       PIC S9(9) BINARY  VALUE 9.
  10 MQIA-INHIBIT-PUT       PIC S9(9) BINARY  VALUE 10.
  10 MQIA-LAST              PIC S9(9) BINARY  VALUE 2000.
  10 MQIA-MAX-MSG-LENGTH    PIC S9(9) BINARY  VALUE 13.
  10 MQIA-MAX-Q-DEPTH       PIC S9(9) BINARY  VALUE 15.
  10 MQIA-OPEN-INPUT-COUNT  PIC S9(9) BINARY  VALUE 17.
  10 MQIA-OPEN-OUTPUT-COUNT PIC S9(9) BINARY  VALUE 18.
  10 MQIA-Q-TYPE            PIC S9(9) BINARY  VALUE 20.
  10 MQIA-SHAREABILITY      PIC S9(9) BINARY  VALUE 23.
  10 MQIA-TRIGGER-CONTROL   PIC S9(9) BINARY  VALUE 24.
  10 MQIA-TRIGGER-TYPE      PIC S9(9) BINARY  VALUE 28.
  10 MQIA-USAGE             PIC S9(9) BINARY  VALUE 12.

**   Integer Attribute Value Denoting 'Not Applicable'
  10 MQIAV-NOT-APPLICABLE PIC S9(9) BINARY  VALUE -1.


*****************************************************************
**  Values Related to MQOPEN Call                              **
*****************************************************************

**   Open Options
  10 MQOO-INPUT-SHARED    PIC S9(9) BINARY  VALUE 2.
  10 MQOO-INPUT-EXCLUSIVE PIC S9(9) BINARY  VALUE 4.
  10 MQOO-BROWSE          PIC S9(9) BINARY  VALUE 8.
  10 MQOO-OUTPUT          PIC S9(9) BINARY  VALUE 16.
  10 MQOO-INQUIRE         PIC S9(9) BINARY  VALUE 32.


*****************************************************************
**  Values Related to All Calls                                **
*****************************************************************

**   String Lengths
  10 MQ-CREATION-DATE-LENGTH     PIC S9(9) BINARY  VALUE 12.
```

```
  10 MQ-CREATION-TIME-LENGTH     PIC S9(9) BINARY  VALUE 8.
  10 MQ-PROCESS-APPL-ID-LENGTH   PIC S9(9) BINARY  VALUE 256.
  10 MQ-PROCESS-DESC-LENGTH      PIC S9(9) BINARY  VALUE 64.
  10 MQ-PROCESS-ENV-DATA-LENGTH  PIC S9(9) BINARY  VALUE 128.
  10 MQ-PROCESS-NAME-LENGTH      PIC S9(9) BINARY  VALUE 48.
  10 MQ-PROCESS-USER-DATA-LENGTH PIC S9(9) BINARY  VALUE 128.
  10 MQ-Q-DESC-LENGTH            PIC S9(9) BINARY  VALUE 64.
  10 MQ-Q-NAME-LENGTH            PIC S9(9) BINARY  VALUE 48.
  10 MQ-Q-MGR-DESC-LENGTH        PIC S9(9) BINARY  VALUE 64.
  10 MQ-Q-MGR-NAME-LENGTH        PIC S9(9) BINARY  VALUE 48.
  10 MQ-TRIGGER-DATA-LENGTH      PIC S9(9) BINARY  VALUE 64.

**   Completion Codes
  10 MQCC-OK      PIC S9(9) BINARY  VALUE 0.
  10 MQCC-WARNING PIC S9(9) BINARY  VALUE 1.
  10 MQCC-FAILED  PIC S9(9) BINARY  VALUE 2.

**   Reason Codes
  10 MQRC-NONE                       PIC S9(9) BINARY  VALUE 0.
  10 MQRC-ACCESS-RESTRICTED          PIC S9(9) BINARY  VALUE 2000.
  10 MQRC-ALIAS-BASE-Q-TYPE-ERROR    PIC S9(9) BINARY  VALUE 2001.
  10 MQRC-ALREADY-CONNECTED          PIC S9(9) BINARY  VALUE 2002.
  10 MQRC-BUFFER-ERROR               PIC S9(9) BINARY  VALUE 2004.
  10 MQRC-BUFFER-LENGTH-ERROR        PIC S9(9) BINARY  VALUE 2005.
  10 MQRC-CHAR-ATTR-LENGTH-ERROR     PIC S9(9) BINARY  VALUE 2006.
  10 MQRC-CHAR-ATTRS-ERROR           PIC S9(9) BINARY  VALUE 2007.
  10 MQRC-CHAR-ATTRS-TOO-SHORT       PIC S9(9) BINARY  VALUE 2008.
  10 MQRC-CONNECTION-BROKEN          PIC S9(9) BINARY  VALUE 2009.
  10 MQRC-DATA-LENGTH-ERROR          PIC S9(9) BINARY  VALUE 2010.
  10 MQRC-EXPIRY-ERROR               PIC S9(9) BINARY  VALUE 2013.
  10 MQRC-FEEDBACK-ERROR             PIC S9(9) BINARY  VALUE 2014.
  10 MQRC-GET-INHIBITED              PIC S9(9) BINARY  VALUE 2016.
  10 MQRC-HANDLE-NOT-AVAILABLE       PIC S9(9) BINARY  VALUE 2017.
  10 MQRC-HCONN-ERROR                PIC S9(9) BINARY  VALUE 2018.
  10 MQRC-HOBJ-ERROR                 PIC S9(9) BINARY  VALUE 2019.
  10 MQRC-INT-ATTR-COUNT-ERROR       PIC S9(9) BINARY  VALUE 2021.
  10 MQRC-INT-ATTR-COUNT-TOO-SMALL   PIC S9(9) BINARY  VALUE 2022.
  10 MQRC-INT-ATTRS-ARRAY-ERROR      PIC S9(9) BINARY  VALUE 2023.
  10 MQRC-MAX-CONNS-LIMIT-REACHED    PIC S9(9) BINARY  VALUE 2025.
  10 MQRC-MD-ERROR                   PIC S9(9) BINARY  VALUE 2026.
  10 MQRC-MISSING-REPLY-TO-Q         PIC S9(9) BINARY  VALUE 2027.
  10 MQRC-MSG-TYPE-ERROR             PIC S9(9) BINARY  VALUE 2029.
  10 MQRC-MSG-TOO-BIG-FOR-Q          PIC S9(9) BINARY  VALUE 2030.
  10 MQRC-NO-MSG-AVAILABLE           PIC S9(9) BINARY  VALUE 2033.
  10 MQRC-NO-MSG-UNDER-CURSOR        PIC S9(9) BINARY  VALUE 2034.
  10 MQRC-NOT-AUTHORIZED             PIC S9(9) BINARY  VALUE 2035.
  10 MQRC-NOT-OPEN-FOR-BROWSE        PIC S9(9) BINARY  VALUE 2036.
  10 MQRC-NOT-OPEN-FOR-INPUT         PIC S9(9) BINARY  VALUE 2037.
  10 MQRC-NOT-OPEN-FOR-INQUIRE       PIC S9(9) BINARY  VALUE 2038.
  10 MQRC-NOT-OPEN-FOR-OUTPUT        PIC S9(9) BINARY  VALUE 2039.
  10 MQRC-OBJECT-CHANGED             PIC S9(9) BINARY  VALUE 2041.
  10 MQRC-OBJECT-IN-USE              PIC S9(9) BINARY  VALUE 2042.
  10 MQRC-OBJECT-TYPE-ERROR          PIC S9(9) BINARY  VALUE 2043.
  10 MQRC-OD-ERROR                   PIC S9(9) BINARY  VALUE 2044.
  10 MQRC-OPTION-NOT-VALID-FOR-TYPE PIC S9(9) BINARY  VALUE 2045.
  10 MQRC-OPTIONS-ERROR              PIC S9(9) BINARY  VALUE 2046.
  10 MQRC-PERSISTENCE-ERROR          PIC S9(9) BINARY  VALUE 2047.
  10 MQRC-PRIORITY-EXCEEDS-MAXIMUM  PIC S9(9) BINARY  VALUE 2049.
  10 MQRC-PRIORITY-ERROR             PIC S9(9) BINARY  VALUE 2050.
  10 MQRC-PUT-INHIBITED              PIC S9(9) BINARY  VALUE 2051.
  10 MQRC-Q-FULL                     PIC S9(9) BINARY  VALUE 2053.
  10 MQRC-Q-SPACE-NOT-AVAILABLE      PIC S9(9) BINARY  VALUE 2056.
  10 MQRC-Q-MGR-NAME-ERROR           PIC S9(9) BINARY  VALUE 2058.
  10 MQRC-Q-MGR-NOT-AVAILABLE        PIC S9(9) BINARY  VALUE 2059.
  10 MQRC-REPORT-OPTIONS-ERROR       PIC S9(9) BINARY  VALUE 2061.
  10 MQRC-SECURITY-ERROR             PIC S9(9) BINARY  VALUE 2063.
  10 MQRC-SELECTOR-COUNT-ERROR       PIC S9(9) BINARY  VALUE 2065.
```

```
  10 MQRC-SELECTOR-LIMIT-EXCEEDED   PIC S9(9) BINARY  VALUE 2066.
  10 MQRC-SELECTOR-ERROR            PIC S9(9) BINARY  VALUE 2067.
  10 MQRC-SELECTOR-NOT-FOR-TYPE     PIC S9(9) BINARY  VALUE 2068.
  10 MQRC-SIGNAL-OUTSTANDING        PIC S9(9) BINARY  VALUE 2069.
  10 MQRC-SIGNAL-REQUEST-ACCEPTED   PIC S9(9) BINARY  VALUE 2070.
  10 MQRC-STORAGE-NOT-AVAILABLE     PIC S9(9) BINARY  VALUE 2071.
  10 MQRC-SYNCPOINT-NOT-AVAILABLE   PIC S9(9) BINARY  VALUE 2072.
  10 MQRC-TRUNCATED-MSG-ACCEPTED    PIC S9(9) BINARY  VALUE 2079.
  10 MQRC-TRUNCATED-MSG-FAILED      PIC S9(9) BINARY  VALUE 2080.
  10 MQRC-UNEXPECTED-CONNECT-ERROR  PIC S9(9) BINARY  VALUE 2081.
  10 MQRC-UNKNOWN-ALIAS-BASE-Q      PIC S9(9) BINARY  VALUE 2082.
  10 MQRC-UNKNOWN-OBJECT-NAME       PIC S9(9) BINARY  VALUE 2085.
  10 MQRC-UNKNOWN-OBJECT-Q-MGR      PIC S9(9) BINARY  VALUE 2086.
  10 MQRC-UNKNOWN-REMOTE-Q-MGR      PIC S9(9) BINARY  VALUE 2087.
  10 MQRC-WAIT-INTERVAL-ERROR       PIC S9(9) BINARY  VALUE 2090.
  10 MQRC-XMIT-Q-TYPE-ERROR         PIC S9(9) BINARY  VALUE 2091.
  10 MQRC-XMIT-Q-USAGE-ERROR        PIC S9(9) BINARY  VALUE 2092.

  10 MQRC-FORMAT-ERROR              PIC S9(9) BINARY  VALUE 2110.
  10 MQRC-SOURCE-CCSID-ERROR        PIC S9(9) BINARY  VALUE 2111.
  10 MQRC-SOURCE-INTEGER-ENC-ERROR  PIC S9(9) BINARY  VALUE 2112.
  10 MQRC-SOURCE-DECIMAL-ENC-ERROR  PIC S9(9) BINARY  VALUE 2113.
  10 MQRC-SOURCE-FLOAT-ENC-ERROR    PIC S9(9) BINARY  VALUE 2114.
  10 MQRC-TARGET-CCSID-ERROR        PIC S9(9) BINARY  VALUE 2115.
  10 MQRC-TARGET-INTEGER-ENC-ERROR  PIC S9(9) BINARY  VALUE 2116.
  10 MQRC-TARGET-DECIMAL-ENC-ERROR  PIC S9(9) BINARY  VALUE 2117.
  10 MQRC-TARGET-FLOAT-ENC-ERROR    PIC S9(9) BINARY  VALUE 2118.
  10 MQRC-NOT-CONVERTED             PIC S9(9) BINARY  VALUE 2119.
  10 MQRC-CONVERTED-MSG-TOO-BIG     PIC S9(9) BINARY  VALUE 2120.
  10 MQRC-SOURCE-LENGTH-ERROR       PIC S9(9) BINARY  VALUE 2143.
  10 MQRC-TARGET-LENGTH-ERROR       PIC S9(9) BINARY  VALUE 2144.
  10 MQRC-DBCS-ERROR                PIC S9(9) BINARY  VALUE 2150.
  10 MQRC-CONVERTED-STRING-TOO-BIG  PIC S9(9) BINARY  VALUE 2190.

  10 MQRC-PMO-ERROR                 PIC S9(9) BINARY  VALUE 2173.
  10 MQRC-GMO-ERROR                 PIC S9(9) BINARY  VALUE 2186.

  10 MQRC-UNEXPECTED-ERROR          PIC S9(9) BINARY  VALUE 2195.
  10 MQRC-MSG-ID-ERROR              PIC S9(9) BINARY  VALUE 2206.
  10 MQRC-CORREL-ID-ERROR           PIC S9(9) BINARY  VALUE 2207.

  10 MQRC-FILE-SYSTEM-ERROR         PIC S9(9) BINARY  VALUE 2208.
  10 MQRC-NO-MSG-LOCKED             PIC S9(9) BINARY  VALUE 2209.


**********************************************************************
**  Values Related to Queue Attributes                              **
**********************************************************************

**   Queue Types
  10 MQQT-LOCAL  PIC S9(9) BINARY  VALUE 1.
  10 MQQT-ALIAS  PIC S9(9) BINARY  VALUE 3.
  10 MQQT-REMOTE PIC S9(9) BINARY  VALUE 6.

**   Queue Definition Types
  10 MQQDT-PREDEFINED PIC S9(9) BINARY  VALUE 1.

**   Inhibit Get
  10 MQQA-GET-INHIBITED PIC S9(9) BINARY  VALUE 1.
  10 MQQA-GET-ALLOWED   PIC S9(9) BINARY  VALUE 0.

**   Inhibit Put
  10 MQQA-PUT-INHIBITED PIC S9(9) BINARY  VALUE 1.
  10 MQQA-PUT-ALLOWED   PIC S9(9) BINARY  VALUE 0.

**   Queue Shareability
  10 MQQA-SHAREABLE     PIC S9(9) BINARY  VALUE 1.
```

```
  10 MQQA-NOT-SHAREABLE PIC S9(9) BINARY  VALUE 0.

**   Message Delivery Sequence
  10 MQMDS-FIFO PIC S9(9) BINARY  VALUE 1.

**   Trigger Control
  10 MQTC-OFF PIC S9(9) BINARY  VALUE 0.
  10 MQTC-ON  PIC S9(9) BINARY  VALUE 1.

**   Trigger Types
  10 MQTT-NONE  PIC S9(9) BINARY  VALUE 0.
  10 MQTT-FIRST PIC S9(9) BINARY  VALUE 1.
  10 MQTT-EVERY PIC S9(9) BINARY  VALUE 2.

**   Queue Usage
  10 MQUS-NORMAL       PIC S9(9) BINARY  VALUE 0.
  10 MQUS-TRANSMISSION PIC S9(9) BINARY  VALUE 1.


****************************************************************
**  Values Related to Process-Definition Attributes            **
****************************************************************

**   Application Type
  10 MQAT-USER-FIRST PIC S9(9) BINARY  VALUE 65536.
  10 MQAT-USER-LAST  PIC S9(9) BINARY  VALUE 999999999.


*
  10 MQAT-OS2        PIC S9(9) BINARY VALUE 4.
  10 MQAT-DOS        PIC S9(9) BINARY VALUE 5.
  10 MQAT-AIX        PIC S9(9) BINARY VALUE 6.
  10 MQAT-OS400      PIC S9(9) BINARY VALUE 8.
  10 MQAT-WINDOWS    PIC S9(9) BINARY VALUE 9.
  10 MQAT-CICS-VSE   PIC S9(9) BINARY VALUE 10.
  10 MQAT-VMS        PIC S9(9) BINARY VALUE 12.
  10 MQAT-GUARDIAN   PIC S9(9) BINARY VALUE 13.
  10 MQAT-VOS        PIC S9(9) BINARY VALUE 14.


****************************************************************
**  Values Related to Queue-Manager Attributes                 **
****************************************************************

**   Syncpoint Availability
  10 MQSP-AVAILABLE PIC S9(9) BINARY  VALUE 1.

  EJECT
*-------------------------------------------------------------*
* COMMON PARMS
 01  FILLER          PIC X(8) VALUE 'PARMS:--'.
 01  WS-HCONN-ADDR-AREA.
     05  WS-HCONN-VALUE             USAGE POINTER.

 01  WS-HOBJ-ADDR-AREA.
     05  WS-HOBJ-VALUE              USAGE POINTER.

 01  WS-HOBJ-ADDR-AREA-REPLY.
     05  WS-HOBJ-VALUE-REPLY        USAGE POINTER.

 01  WS-CCODE-ADDR-AREA.
     05  WS-CCODE-VALUE             PIC S9(8) COMP.

 01  WS-RCODE-ADDR-AREA.
     05  WS-RCODE-VALUE             PIC S9(8) COMP.

*-------------------------------------------------------------*
```

```
*--CONNECT PARM
 01  WS-QM-NAME-AREA.
     05  WS-QM-NAME-CONNECT         PIC  X(48).

*--OPEN    PARM
 01  WS-Q-NAME-AREA.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQODV                                   **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQOD structure       **
**                                                             **
**  VERSION 2.1.0                                              **
**                                                             **
**  FUNCTION:         This file declares the MQOD structure,   **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************

**   MQOD structure
  10 MQOD.
**    Structure identifier
    15 MQOD-STRUCID         PIC X(4) VALUE 'OD  '.
**    Structure version number
    15 MQOD-VERSION         PIC S9(9) BINARY   VALUE 1.
**    Object type
    15 MQOD-OBJECTTYPE      PIC S9(9) BINARY   VALUE 1.
**    Object name
    15 MQOD-OBJECTNAME      PIC X(48) VALUE SPACES.
**    Object queue manager name
    15 MQOD-OBJECTQMGRNAME  PIC X(48) VALUE SPACES.
**    Dynamic queue name
    15 MQOD-DYNAMICQNAME    PIC X(48) VALUE '*'.
**    Alternate user identifier
    15 MQOD-ALTERNATEUSERID PIC X(12) VALUE SPACES.

  01  WS-Q-OPEN-OPTIONS.
      05  WS-Q-OPEN-OPTIONS-VALUE    PIC S9(8) COMP.
      EJECT

*--INQ

  01  MQI-CHAR-ATTR-LENGTH.
      05  WS-CHAR-ATTR-LENGTH        PIC S9(8) COMP.
  01  MQI-CHAR-ATTR.
      05  WS-CHAR-ATTR            PIC X(500) VALUE SPACES.
  01  WS-D-FILLER REDEFINES MQI-CHAR-ATTR.
      05  WS-A-DESC               PIC X(64).
      05  WS-A-Q-N                PIC X(64).



*--PUT/GET PARM
 01  WS-MSG-DESCRIPTOR.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQMDV                                   **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQMD structure       **
**                                                             **
**  VERSION 2.1.0                                              **
**                                                             **
**  FUNCTION:         This file declares the MQMD structure,   **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
```

```
*****************************************************************

**   MQMD structure
  10 MQMD.
**    Structure identifier
   15 MQMD-STRUCID           PIC X(4) VALUE 'MD  '.
**    Structure version number
   15 MQMD-VERSION           PIC S9(9) BINARY  VALUE 1.
**    Reserved
   15 MQMD-REPORT            PIC S9(9) BINARY  VALUE 0.
**    Message type
   15 MQMD-MSGTYPE           PIC S9(9) BINARY  VALUE 8.
**    Reserved
   15 MQMD-EXPIRY            PIC S9(9) BINARY  VALUE -1.
**    Feedback code
   15 MQMD-FEEDBACK          PIC S9(9) BINARY  VALUE 0.
**    Data encoding
   15 MQMD-ENCODING          PIC S9(9) BINARY  VALUE 785.
**    Coded character set identifier
   15 MQMD-CODEDCHARSETID    PIC S9(9) BINARY  VALUE 0.
**    Format name
   15 MQMD-FORMAT            PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-PRIORITY          PIC S9(9) BINARY  VALUE 0.
**    Message persistence
   15 MQMD-PERSISTENCE       PIC S9(9) BINARY  VALUE 2.
**    Message identifier
   15 MQMD-MSGID             PIC X(24) VALUE LOW-VALUES.
**    Correlation identifier
   15 MQMD-CORRELID          PIC X(24) VALUE LOW-VALUES.
**    Reserved
   15 MQMD-BACKOUTCOUNT      PIC S9(9) BINARY  VALUE 0.
**    Name of reply queue
   15 MQMD-REPLYTOQ          PIC X(48) VALUE SPACES.
**    Name of reply queue manager
   15 MQMD-REPLYTOQMGR       PIC X(48) VALUE SPACES.
**    Reserved
   15 MQMD-USERIDENTIFIER    PIC X(12) VALUE SPACES.
**    Reserved
   15 MQMD-ACCOUNTINGTOKEN   PIC X(32) VALUE LOW-VALUES.
**    Reserved
   15 MQMD-APPLIDENTITYDATA  PIC X(32) VALUE SPACES.
**    Reserved
   15 MQMD-PUTAPPLTYPE       PIC S9(9) BINARY  VALUE 0.
**    Reserved
   15 MQMD-PUTAPPLNAME       PIC X(28) VALUE SPACES.
**    Reserved
   15 MQMD-PUTDATE           PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-PUTTIME           PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-APPLORIGINDATA    PIC X(4) VALUE SPACES.


 01  WS-PUT-OPTIONS.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQPMOV                                  **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQPMO structure      **
**                                                             **
**  VERSION 2.1.0                                              **
**                                                             **
**  FUNCTION:         This file declares the MQPMO structure,  **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
```

```
*****************************************************************

**   MQPMO structure
  10 MQPMO.
**    Structure identifier
   15 MQPMO-STRUCID           PIC X(4) VALUE 'PMO '.
**    Structure version number
   15 MQPMO-VERSION           PIC S9(9) BINARY   VALUE 1.
**    Reserved
   15 MQPMO-OPTIONS           PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-TIMEOUT           PIC S9(9) BINARY   VALUE -1.
**    Reserved
   15 MQPMO-CONTEXT           PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-KNOWNDESTCOUNT    PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-UNKNOWNDESTCOUNT  PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-INVALIDDESTCOUNT  PIC S9(9) BINARY   VALUE 0.
**    Resolved name of destination queue
   15 MQPMO-RESOLVEDQNAME     PIC X(48) VALUE SPACES.
**    Resolved name of destination queue manager
   15 MQPMO-RESOLVEDQMGRNAME  PIC X(48) VALUE SPACES.


 01  WS-GET-OPTIONS.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQGMOV                                  **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQGMO structure      **
**                                                             **
**  VERSION 2.1.0                                            **
**                                                             **
**  FUNCTION:         This file declares the MQGMO structure,  **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************

**   MQGMO structure
  10 MQGMO.
**    Structure identifier
   15 MQGMO-STRUCID        PIC X(4) VALUE 'GMO '.
**    Structure version number
   15 MQGMO-VERSION        PIC S9(9) BINARY   VALUE 1.
**    Options
   15 MQGMO-OPTIONS        PIC S9(9) BINARY   VALUE 0.
**    Wait interval
   15 MQGMO-WAITINTERVAL   PIC S9(9) BINARY   VALUE 0.
**    Signal
   15 MQGMO-SIGNAL1        PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQGMO-SIGNAL2        PIC S9(9) BINARY   VALUE 0.
**    Resolved name of destination queue
   15 MQGMO-RESOLVEDQNAME  PIC X(48) VALUE SPACES.


 01  WS-DATA-L-AREA.
     05  WS-DATA-LENGTH-USER        PIC S9(8) COMP VALUE +200.

 01  WS-BUFFER-L-AREA.
     05  WS-BUFFER-LENGTH           PIC S9(8) COMP VALUE +200.

 77  WS-MSG-LENGTH                  PIC S9(8) COMP VALUE +200.
 01  WS-RECEIVE-BUFFER              PIC X(1024) VALUE SPACE.
```

TTPTST2.Z

```
 01  WS-REMAINDER                  PIC X(1024) VALUE SPACES.
 01  WS-POINTER                    PIC 9(4).
 01  WS-MSG-AREA.
     05  FILLER                    PIC  X(500) VALUE
     'THIS IS A MESSAGE TEXT'.

 01  WS-BUFFER-AREA.
     05  WS-BUFFER-TS              PIC  X(16)  VALUE SPACES.
     05  WS-BUFFER-TEXT            PIC  X(500) VALUE SPACES.

*    COPY    MQWEOWS.
*-----------------------------------------------------------------*
 LINKAGE SECTION.
*-----------------------------------------------------------------*

 01  LK-DATA.
     05  FILLER                   PIC X.
  EJECT
*-----------------------------------------------------------------*
 PROCEDURE DIVISION.
*-----------------------------------------------------------------*

 0000-MAIN-LINE.

*--INITIALIZE
     MOVE  'INIT   ' TO   WS-LEVEL.
     PERFORM 1000-INITIALIZE
        THRU 1000-EXIT.

*-----------------------------------------------------------------*
     PERFORM    1   TIMES

*--SEND QUEUE RECORDS
          IF WS-PUT  OR WS-BOTH
           THEN
               PERFORM 2000-PUT-MESSAGES
                  THRU 2000-EXIT
          END-IF
*--GET  QUEUE RECORDS
          IF WS-GET  OR WS-BOTH
           THEN
               PERFORM 3000-GET-MESSAGES
                  THRU 3000-EXIT
          END-IF

          IF WS-PUT1
           THEN
               PERFORM 4000-PUT1-MESSAGES
                  THRU 4000-EXIT
          END-IF

          IF WS-GET-WITH-DELETE
           THEN
               PERFORM 5000-GETD-MESSAGES
                  THRU 5000-EXIT
          END-IF

          IF WS-PUT-WITH-REPLY
           THEN
               PERFORM 6000-PUT-WITH-REPLY
                  THRU 6000-EXIT
          END-IF
          IF WS-INQ
           THEN
               PERFORM 6500-INQ-MESSAGES
                  THRU 6500-EXIT
          END-IF
```

```
     END-PERFORM.
*---------------------------------------------------------------*
 0000-SEND-TOTALS.
     IF NOT WS-STARTED
      THEN
             PERFORM   7000-SEND-TOTALS.


*---------------------------------------------------------------*
 0000-RETURN.
     EXEC CICS RETURN
          END-EXEC.

     GOBACK.
  EJECT
*---------------------------------------------------------------*
 1000-INITIALIZE.
*---------------------------------------------------------------*
* PURPOSE: SETUP DATA AREAS
*---------------------------------------------------------------*
*--GET STARTED CICS CODE
      EXEC CICS ASSIGN
                STARTCODE (WS-STARTCODE)
           END-EXEC.
*
*--SET TIME/DATE
     EXEC  CICS ASKTIME
                ABSTIME(WS-ABSTIME)
            END-EXEC.
*
     EXEC  CICS FORMATTIME
                ABSTIME(WS-ABSTIME)
                YYMMDD (WS-DATE-YYMMDD)
            END-EXEC.

     IF WS-DATE-YY >  50
      THEN
             MOVE  19                  TO  WS-DATE-CC
      ELSE
             MOVE  20                  TO  WS-DATE-CC.
*
     MOVE  EIBTIME   TO   WS-TIME-9.
     EXEC CICS FORMATTIME
               ABSTIME (WS-ABSTIME)
               MMDDYY  (WS-FORMATTED-DATE)
               DATESEP ('/')
               TIME    (WS-FORMATTED-TIME)
               TIMESEP (':')
          END-EXEC.
*
*--GET INPUT INFO...
     IF  START-WITH-DATA
      THEN
             PERFORM  1100-PASSED-INFO
      ELSE
             PERFORM  1200-SETUP-INPUT.

*---------------------------------------------------------------*
 1000-EXIT.
     EXIT.

  EJECT
*---------------------------------------------------------------*
 1100-PASSED-INFO.
*---------------------------------------------------------------*
```

```
* PURPOSE: SETUP PASSED DATA AREAS
*-----------------------------------------------------------------*
*--GET PASSED DATA
     MOVE LENGTH OF WS-PASSED-INFO TO WS-PASS-MSG-LENGTH.
     EXEC CICS RETRIEVE
                    INTO (WS-PASSED-INFO)
                  LENGTH (WS-PASS-MSG-LENGTH)
            END-EXEC.

     IF  WS-PASS-MSG-LENGTH <  LENGTH OF WS-PASSED-INFO
      THEN
             GO TO  0000-RETURN.
*
     SET WS-STARTED  TO TRUE.
     MOVE TST2-FUNCTION    TO WS-DATA-FUNCTION.
     MOVE TST2-PUT-NUM-MSG TO  WS-PROCESS-TIMES.
     MOVE TST2-PUT-QUEUE-NAME TO WS-DATA-QUEUE.
     MOVE TST2-PUT-MSG-SIZE   TO WS-MSG-LENGTH.
     MOVE TST2-PUT-MSG        TO WS-MSG-AREA.
     MOVE TST2-PUT-MSG-TIMESTAMP  TO WS-TIMESTAMP.

*-----------------------------------------------------------------*
   EJECT
*-----------------------------------------------------------------*
 1100-SEND-HELP.
*-----------------------------------------------------------------*
*--SEND HELPLIST
     EXEC CICS SEND
               FROM   (WS-HELP)
               LENGTH (LENGTH OF WS-HELP)
               ERASE
     END-EXEC.

*-----------------------------------------------------------------*
   EJECT
*-----------------------------------------------------------------*
 1200-SETUP-INPUT.
*-----------------------------------------------------------------*
*--GET DATA
     MOVE LENGTH OF WS-RECEIVE-BUFFER TO WS-DATA-LENGTH
     EXEC  CICS RECEIVE
                     INTO(WS-RECEIVE-BUFFER)
                     LENGTH(WS-DATA-LENGTH)
     END-EXEC
*-- Handle variable number of parameters.
     MOVE +1 TO WS-POINTER
     UNSTRING WS-RECEIVE-BUFFER DELIMITED BY ALL SPACES INTO
              WS-TRANSID
              WS-DATA-FUNCTION
              WITH POINTER WS-POINTER
     END-UNSTRING
*-- Handle mixed case.
     INSPECT WS-DATA-FUNCTION CONVERTING
      'abcdefghijklmnopqrstuvwxyz' TO
      'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

*-- Check for invalid function codes.
     IF (NOT VALID-DATA-FUNCTION) OR WS-HELP-FUN
      THEN
          PERFORM  1100-SEND-HELP
          GO TO   0000-RETURN
     END-IF

*--Extract the remaining options according to the function code.
     IF NOT WS-INQ THEN
       UNSTRING WS-RECEIVE-BUFFER DELIMITED BY ALL SPACES INTO
                WS-DATA-TIMES
```

```
               WS-DATA-QUEUE
               WS-TIMESTAMP
               WITH POINTER WS-POINTER
       END-UNSTRING
     ELSE
       UNSTRING WS-RECEIVE-BUFFER DELIMITED BY ALL SPACES INTO
               WS-DATA-QUEUE
               WITH POINTER WS-POINTER
       END-UNSTRING
     END-IF
*--CHECK WHAT WE'RE DOING
*-- --COMMAND IS  "TST2 GET 01 QUEUENAME"
*
     MOVE  WS-DATA-TIMES  TO WS-PROCESS-TIMES.
     IF WS-PROCESS-TIMES  EQUAL  ZERO
      THEN
            MOVE  100  TO   WS-PROCESS-TIMES.
*
*--IF REPLY ..SEND AND GET
     IF NOT WS-PUT-WITH-REPLY
      THEN
            GO TO  1000-EXIT.
*
*--IF REPLY ..SEND AND GET
     EXEC CICS SEND
               FROM   (WS-NEED-REPLY)
               LENGTH (LENGTH OF WS-NEED-REPLY)
               ERASE
     END-EXEC.

     EXEC  CICS RECEIVE
                     SET (ADDRESS OF LK-DATA)
                  LENGTH(WS-DATA-LENGTH)
          END-EXEC.

     IF WS-DATA-LENGTH >  48
      THEN
             MOVE +48  TO  WS-DATA-LENGTH.

     MOVE WS-DATA-LENGTH TO WS-DATA-FLENGTH
     MOVE  LK-DATA(1:WS-DATA-FLENGTH) TO
                            WS-REPLY-Q(1:WS-DATA-FLENGTH)
     .
*
*---------------------------------------------------------------*
  EJECT
*---------------------------------------------------------------*
 2000-PUT-MESSAGES.
*---------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          PUT
*          CLOSE,    DISCONNECT
*---------------------------------------------------------------*
*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
     MOVE MQCC-OK                        TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE                      TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'MQCONN'  USING  WS-QM-NAME-AREA
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
```

```
               GO TO  9900-ERR-DISPLAY.
*
*--MQOPEN   QUEUE TO QM
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-OUTPUT    TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE SPACES         TO   MQOD-OBJECTQMGRNAME.
     MOVE WS-DATA-QUEUE  TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK        TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE      TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE   TO   NULL.
     CALL 'MQOPEN'  USING  WS-HCONN-ADDR-AREA
                           WS-Q-NAME-AREA
                           WS-Q-OPEN-OPTIONS
                           WS-HOBJ-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
               GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES

*--CHECK IF MUST PUT TIME STAMP ON MESSAGE
       IF WS-PUT-TIMESTAMP
        THEN
             PERFORM 8000-GET-TIME-STAMP
             MOVE WS-COUNT TO WS-TIMESTAMP-COUNT
             MOVE WS-TIMESTAMP-VALUE  TO  WS-BUFFER-TS
             MOVE  LENGTH OF WS-BUFFER-TS
                                  TO WS-BUFFER-LENGTH
             ADD WS-MSG-LENGTH     TO WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA         TO  WS-BUFFER-TEXT
        ELSE
             MOVE WS-MSG-LENGTH       TO  WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA         TO  WS-BUFFER-AREA
       END-IF

*--MQPUT TO QUEUE TO QM
       MOVE  'PUT'     TO   WS-FUNCTION
       MOVE MQCC-OK    TO   WS-CCODE-VALUE
       MOVE MQRC-NONE  TO   WS-RCODE-VALUE
       CALL 'MQPUT'   USING  WS-HCONN-ADDR-AREA
                             WS-HOBJ-ADDR-AREA
                             WS-MSG-DESCRIPTOR
                             WS-PUT-OPTIONS
                             WS-BUFFER-L-AREA
                             WS-BUFFER-AREA
                             WS-CCODE-ADDR-AREA
                             WS-RCODE-ADDR-AREA
*
       IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        THEN
                GO TO  9900-ERR-DISPLAY
       END-IF
       ADD  +1         TO   WS-COUNT


*--SYNPOINT PUT SO ECHO CAN GET IT
*-- --CHECK IF "NEGATIVE " PROCESSING OPTION SPECIFIED
*      IF WS-DATA-SYNC-FLAG NOT EQUAL '-'
*       THEN
*            EXEC  CICS SYNCPOINT
*                  END-EXEC
*      END-IF
```

```
     END-PERFORM.
*-----------------------------------------------------------------*

*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQCLOSE' USING  WS-HCONN-ADDR-AREA
                           WS-HOBJ-ADDR-AREA
                           WS-Q-OPEN-OPTIONS
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQDISC'  USING
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 2000-EXIT.
     EXIT.
  EJECT
*-----------------------------------------------------------------*
 3000-GET-MESSAGES.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          GET
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------*
*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'MQCONN'  USING  WS-QM-NAME-AREA
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*
*--MQOPEN   QUEUE TO QM
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-INPUT-SHARED TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE SPACES            TO   MQOD-OBJECTQMGRNAME.
     MOVE WS-DATA-QUEUE     TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK           TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE         TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE   TO   NULL.
     CALL 'MQOPEN'  USING  WS-HCONN-ADDR-AREA
```

```
                          WS-Q-NAME-AREA
                          WS-Q-OPEN-OPTIONS
                          WS-HOBJ-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.


*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES
*
*--MQGET TO QUEUE TO QM
       MOVE  'GET'     TO   WS-FUNCTION
       MOVE MQCC-OK    TO   WS-CCODE-VALUE
       MOVE MQRC-NONE  TO   WS-RCODE-VALUE
       MOVE 500  TO   WS-BUFFER-LENGTH
       MOVE MQGMO-ACCEPT-TRUNCATED-MSG
                 TO   MQGMO-OPTIONS
       MOVE  LOW-VALUES TO  MQMD-MSGID
                            MQMD-CORRELID
*
       CALL 'MQGET'   USING  WS-HCONN-ADDR-AREA
                             WS-HOBJ-ADDR-AREA
                             WS-MSG-DESCRIPTOR
                             WS-GET-OPTIONS
                             WS-BUFFER-L-AREA
                             WS-BUFFER-AREA
                             WS-DATA-L-AREA
                             WS-CCODE-ADDR-AREA
                             WS-RCODE-ADDR-AREA
*
       IF  (WS-CCODE-VALUE   NOT EQUAL ZERO)
        THEN
             IF WS-RCODE-VALUE EQUAL 2079
              THEN
                   SET WS-TRUNCATED-MESSAGES  TO TRUE
              ELSE
             IF WS-RCODE-VALUE EQUAL 2033
              THEN
                   SET WS-END-OF-MESSAGES  TO TRUE
                   GO TO     3000-GET-EOF
              ELSE
                  GO TO  9900-ERR-DISPLAY
             END-IF
             END-IF
*
       END-IF
*-- --CHECK IF "NEGATIVE " PROCESSING OPTION SPECIFIED
       IF WS-DATA-SYNC-FLAG  NOT EQUAL  '-'
        THEN
             EXEC  CICS SYNCPOINT
                   END-EXEC
       END-IF
       ADD  +1         TO   WS-COUNT

     END-PERFORM.
*-----------------------------------------------------------------*
*
 3000-GET-EOF.
*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
```

```
     CALL 'MQCLOSE' USING  WS-HCONN-ADDR-AREA
                           WS-HOBJ-ADDR-AREA
                           WS-Q-OPEN-OPTIONS
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQDISC'  USING
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 3000-EXIT.
     EXIT.
  EJECT
*-----------------------------------------------------------------*
 4000-PUT1-MESSAGES.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          PUT
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------*
*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
     MOVE MQCC-OK                       TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE                     TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'MQCONN'  USING  WS-QM-NAME-AREA
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES
*
*--CHECK IF MUST PUT TIME STAMP ON MESSAGE
       IF WS-PUT-TIMESTAMP
        THEN
             PERFORM 8000-GET-TIME-STAMP
             MOVE WS-TIMESTAMP-VALUE  TO  WS-BUFFER-TS
             MOVE  LENGTH OF WS-BUFFER-TS
                                   TO WS-BUFFER-LENGTH
             ADD WS-MSG-LENGTH     TO WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA         TO  WS-BUFFER-TEXT
        ELSE
             MOVE WS-MSG-LENGTH       TO  WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA         TO  WS-BUFFER-AREA
       END-IF
*
```

```
*--MQPUT1   QUEUE TO QM
       MOVE  'PUT1'    TO   WS-FUNCTION
       MOVE MQOO-OUTPUT    TO   MQPMO-OPTIONS
       MOVE SPACES         TO   MQOD-OBJECTQMGRNAME
       MOVE WS-DATA-QUEUE  TO   MQOD-OBJECTNAME
       MOVE MQCC-OK        TO   WS-CCODE-VALUE
       MOVE MQRC-NONE      TO   WS-RCODE-VALUE
       CALL 'MQPUT1'  USING  WS-HCONN-ADDR-AREA
                             WS-Q-NAME-AREA
                             WS-MSG-DESCRIPTOR
                             WS-PUT-OPTIONS
                             WS-BUFFER-L-AREA
                             WS-BUFFER-AREA
                             WS-CCODE-ADDR-AREA
                             WS-RCODE-ADDR-AREA
*
       IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        THEN
                   GO TO  9900-ERR-DISPLAY
       END-IF
*
     END-PERFORM.
*-----------------------------------------------------------------*
*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQDISC'  USING
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 4000-EXIT.
     EXIT.
  EJECT
*-----------------------------------------------------------------*
 5000-GETD-MESSAGES.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          GET
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------*
*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'MQCONN'  USING  WS-QM-NAME-AREA
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*
*--MQOPEN   QUEUE TO QM
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-BROWSE       TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE SPACES            TO   MQOD-OBJECTQMGRNAME.
```

```
     MOVE WS-DATA-QUEUE     TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK           TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE         TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE   TO   NULL.
     CALL 'MQOPEN'  USING  WS-HCONN-ADDR-AREA
                           WS-Q-NAME-AREA
                           WS-Q-OPEN-OPTIONS
                           WS-HOBJ-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*

*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES
*
*--MQGET TO QUEUE TO QM
       MOVE  'GET'     TO   WS-FUNCTION
       MOVE MQCC-OK    TO   WS-CCODE-VALUE
       MOVE MQRC-NONE  TO   WS-RCODE-VALUE
       MOVE 500  TO   WS-BUFFER-LENGTH
       MOVE MQGMO-BROWSE-FIRST   TO  MQGMO-OPTIONS
       ADD  MQGMO-ACCEPT-TRUNCATED-MSG
                              TO  MQGMO-OPTIONS
       MOVE  LOW-VALUES TO  MQMD-MSGID
                            MQMD-CORRELID
*
       CALL 'MQGET'   USING  WS-HCONN-ADDR-AREA
                             WS-HOBJ-ADDR-AREA
                             WS-MSG-DESCRIPTOR
                             WS-GET-OPTIONS
                             WS-BUFFER-L-AREA
                             WS-BUFFER-AREA
                             WS-DATA-L-AREA
                             WS-CCODE-ADDR-AREA
                             WS-RCODE-ADDR-AREA

*-- --CHECK RC
       IF  (WS-CCODE-VALUE   NOT EQUAL ZERO)
        THEN
             IF WS-RCODE-VALUE EQUAL 2079
              THEN
                   SET WS-TRUNCATED-MESSAGES  TO TRUE
              ELSE
             IF WS-RCODE-VALUE EQUAL 2033
              THEN
                   SET WS-END-OF-MESSAGES  TO TRUE
                   GO TO  5000-GET-EOF
              ELSE
                  GO TO  9900-ERR-DISPLAY
             END-IF
             END-IF
       END-IF
*
       MOVE ZERO TO MQMD-REPORT

*--MQGET TO QUEUE TO QM W/ DELETE UNDER CURSOR
       MOVE  'GET'     TO   WS-FUNCTION
       MOVE MQCC-OK    TO   WS-CCODE-VALUE
       MOVE MQRC-NONE  TO   WS-RCODE-VALUE
       MOVE MQGMO-MSG-UNDER-CURSOR   TO  MQGMO-OPTIONS
       MOVE 500  TO   WS-BUFFER-LENGTH
       MOVE  LOW-VALUES TO  MQMD-MSGID
```

```
                          MQMD-CORRELID
*
       CALL 'MQGET'   USING  WS-HCONN-ADDR-AREA
                             WS-HOBJ-ADDR-AREA
                             WS-MSG-DESCRIPTOR
                             WS-GET-OPTIONS
                             WS-BUFFER-L-AREA
                             WS-BUFFER-AREA
                             WS-DATA-L-AREA
                             WS-CCODE-ADDR-AREA
                             WS-RCODE-ADDR-AREA
*
       IF  (WS-CCODE-VALUE   NOT EQUAL ZERO)
        THEN
             IF WS-RCODE-VALUE EQUAL 2079
              THEN
                   SET WS-TRUNCATED-MESSAGES  TO TRUE
              ELSE
                  GO TO  9900-ERR-DISPLAY
             END-IF
       END-IF
*

*--ADDED 4/ 5/93
*-- --CHECK IF "NEGATIVE " PROCESSING OPTION SPECIFIED
       IF WS-DATA-SYNC-FLAG  NOT EQUAL  '-'
        THEN
             EXEC  CICS SYNCPOINT
                   END-EXEC
       END-IF
       ADD  +1         TO   WS-COUNT

     END-PERFORM.
*-----------------------------------------------------------------*
 5000-GET-EOF.
*
*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQCLOSE' USING  WS-HCONN-ADDR-AREA
                           WS-HOBJ-ADDR-AREA
                           WS-Q-OPEN-OPTIONS
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQDISC'  USING
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 5000-EXIT.
     EXIT.
```

```
  EJECT
*-----------------------------------------------------------------*
 6000-PUT-WITH-REPLY.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          PUT
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------*
*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
     MOVE MQCC-OK                       TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE                     TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'MQCONN'  USING  WS-QM-NAME-AREA
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*

*--MQOPEN   QUEUE FOR REPLY QUEUE
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-INPUT-SHARED TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE  MQMT-REQUEST TO  MQMD-MSGTYPE.
     MOVE  SPACES       TO  MQMD-REPLYTOQMGR.
     MOVE  SPACES       TO  MQMD-REPLYTOQ.

     MOVE SPACES            TO   MQOD-OBJECTQMGRNAME.
     MOVE  WS-REPLY-Q       TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK           TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE         TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE-REPLY   TO   NULL.
     CALL 'MQOPEN'  USING  WS-HCONN-ADDR-AREA
                           WS-Q-NAME-AREA
                           WS-Q-OPEN-OPTIONS
                           WS-HOBJ-ADDR-AREA-REPLY
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.


*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES
*--CHECK IF MUST PUT TIME STAMP ON MESSAGE
       IF WS-PUT-TIMESTAMP
        THEN
             PERFORM 8000-GET-TIME-STAMP
             MOVE WS-TIMESTAMP-VALUE  TO  WS-BUFFER-TS
             ADD WS-MSG-LENGTH  LENGTH OF WS-BUFFER-TS
                                GIVING WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA         TO  WS-BUFFER-TEXT
        ELSE
             MOVE WS-MSG-LENGTH       TO  WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA         TO  WS-BUFFER-AREA
       END-IF
*
*--MQPUT1   QUEUE TO QM
       MOVE  'PUT1'    TO   WS-FUNCTION
       MOVE  MQMT-REPLY   TO  MQMD-MSGTYPE
```

```
           MOVE  SPACES       TO  MQMD-REPLYTOQMGR
           MOVE  WS-REPLY-Q   TO  MQMD-REPLYTOQ

           MOVE MQOO-OUTPUT    TO   MQPMO-OPTIONS
           MOVE SPACES         TO   MQOD-OBJECTQMGRNAME
           MOVE WS-DATA-QUEUE  TO   MQOD-OBJECTNAME
           MOVE MQCC-OK        TO   WS-CCODE-VALUE
           MOVE MQRC-NONE      TO   WS-RCODE-VALUE
           CALL 'MQPUT1'  USING  WS-HCONN-ADDR-AREA
                                 WS-Q-NAME-AREA
                                 WS-MSG-DESCRIPTOR
                                 WS-PUT-OPTIONS
                                 WS-BUFFER-L-AREA
                                 WS-BUFFER-AREA
                                 WS-CCODE-ADDR-AREA
                                 WS-RCODE-ADDR-AREA
      *
           IF WS-CCODE-VALUE   NOT EQUAL  ZERO
            THEN
                    GO TO  9900-ERR-DISPLAY
           END-IF

      *--SYNPOINT PUT SO ECHO CAN GET IT
           EXEC  CICS SYNCPOINT
                 END-EXEC

      *--MQGET TO QUEUE TO QM
           MOVE  'GET'     TO   WS-FUNCTION
           MOVE  MQMT-REQUEST TO  MQMD-MSGTYPE
           MOVE  LOW-VALUES    TO  MQMD-MSGID
                                   MQMD-CORRELID
           MOVE  SPACES       TO  MQMD-REPLYTOQMGR

           MOVE  SPACES       TO  MQMD-REPLYTOQ

           MOVE MQCC-OK    TO   WS-CCODE-VALUE
           MOVE MQRC-NONE  TO   WS-RCODE-VALUE
           MOVE 500  TO   WS-BUFFER-LENGTH
           MOVE MQGMO-ACCEPT-TRUNCATED-MSG
                                    TO  MQGMO-OPTIONS
           ADD  MQGMO-WAIT
                     TO   MQGMO-OPTIONS

           MOVE  LOW-VALUES TO  MQMD-MSGID
                                MQMD-CORRELID
      *--WAIT 30 SECONDS (IE, 30,000 MILL-SECONDS)
           MOVE +30000  TO   MQGMO-WAITINTERVAL
      *
           CALL 'MQGET'   USING  WS-HCONN-ADDR-AREA
                                 WS-HOBJ-ADDR-AREA-REPLY
                                 WS-MSG-DESCRIPTOR
                                 WS-GET-OPTIONS
                                 WS-BUFFER-L-AREA
                                 WS-BUFFER-AREA
                                 WS-DATA-L-AREA
                                 WS-CCODE-ADDR-AREA
                                 WS-RCODE-ADDR-AREA
      *
           IF  (WS-CCODE-VALUE   NOT EQUAL ZERO)
            THEN
                 IF WS-RCODE-VALUE EQUAL 2079
                  THEN
                       SET WS-TRUNCATED-MESSAGES  TO TRUE
                  ELSE
                 IF WS-RCODE-VALUE EQUAL 2033
                  THEN
                       SET WS-END-OF-MESSAGES  TO TRUE
```

```
                GO TO  6000-PUT-WITH-EOF
             ELSE
                GO TO  9900-ERR-DISPLAY
           END-IF
           END-IF
      END-IF

      ADD  +1         TO   WS-COUNT

*--SYNPOINT PUT SO ECHO CAN GET IT
      EXEC  CICS SYNCPOINT
            END-EXEC


    END-PERFORM.
*-----------------------------------------------------------------*
 6000-PUT-WITH-EOF.

*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQCLOSE' USING  WS-HCONN-ADDR-AREA
                           WS-HOBJ-ADDR-AREA-REPLY
                           WS-Q-OPEN-OPTIONS
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.


*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQDISC'  USING
                           WS-HCONN-ADDR-AREA
                           WS-CCODE-ADDR-AREA
                           WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 6000-EXIT.
     EXIT.
   EJECT
*-----------------------------------------------------------------*
*-- INQ function added.
 6500-INQ-MESSAGES.
*-----------------------------------------------------------------
* PURPOSE: CONNECT , OPEN
*          INQ
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------
*
*--MQCONNECT TO QM
      MOVE  'CONNECT' TO   WS-FUNCTION.
      MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
      MOVE MQCC-OK                       TO   WS-CCODE-VALUE.
      MOVE MQRC-NONE                     TO   WS-RCODE-VALUE.
      SET WS-HCONN-VALUE  TO NULL.
      CALL 'MQCONN'  USING  WS-QM-NAME-AREA
```

```
                          WS-HCONN-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
      IF WS-CCODE-VALUE   NOT EQUAL  ZERO
       THEN
                GO TO  9900-ERR-DISPLAY.
*
*--MQOPEN   QUEUE TO QM
      MOVE  'OPEN'    TO   WS-FUNCTION.
      MOVE MQOO-INQUIRE   TO   WS-Q-OPEN-OPTIONS-VALUE.
      MOVE SPACES         TO   MQOD-OBJECTQMGRNAME.
      MOVE WS-DATA-QUEUE  TO   MQOD-OBJECTNAME.
      MOVE MQCC-OK        TO   WS-CCODE-VALUE.
      MOVE MQRC-NONE      TO   WS-RCODE-VALUE.
      SET WS-HOBJ-VALUE   TO   NULL.
      CALL 'MQOPEN'  USING  WS-HCONN-ADDR-AREA
                          WS-Q-NAME-AREA
                          WS-Q-OPEN-OPTIONS
                          WS-HOBJ-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
      IF WS-CCODE-VALUE   NOT EQUAL  ZERO
       THEN
                GO TO  9900-ERR-DISPLAY.

*--SETUP INQ PARMS
      MOVE MQCA-Q-DESC   TO MQI-SELECTOR-ENTRY (1).
      MOVE MQCA-Q-NAME   TO MQI-SELECTOR-ENTRY (2).
      MOVE MQIA-INHIBIT-PUT   TO MQI-SELECTOR-ENTRY (3).
      MOVE MQIA-Q-TYPE   TO MQI-SELECTOR-ENTRY (4).
      MOVE MQIA-MAX-MSG-LENGTH   TO MQI-SELECTOR-ENTRY (5).
      MOVE +5      TO WS-SELECTOR-COUNT.
      MOVE LENGTH OF WS-CHAR-ATTR TO
                           WS-CHAR-ATTR-LENGTH
*
*--MQPUT TO QUEUE TO QM
      MOVE  'INQ'     TO   WS-FUNCTION.
      MOVE MQCC-OK    TO   WS-CCODE-VALUE.
      MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
      CALL 'MQINQ'   USING    WS-HCONN-ADDR-AREA
                              WS-HOBJ-ADDR-AREA
                              WS-SELECTOR-COUNT
                              MQI-SELECTOR
                              MQI-IN-ATTR-COUNT
                              MQI-IN-ATTR
                              MQI-CHAR-ATTR-LENGTH
                              MQI-CHAR-ATTR
                              WS-CCODE-ADDR-AREA
                              WS-RCODE-ADDR-AREA.
*
      IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        THEN
              GO TO  9900-ERR-DISPLAY
      ELSE
         STRING 'NAME: ' WS-A-Q-N  DELIMITED BY SIZE
           INTO WS-OK-MSG-7
         END-STRING
         STRING 'DESC: ' WS-A-DESC DELIMITED BY SIZE
           INTO WS-OK-MSG-8
         END-STRING
         MOVE MQI-IN-ATTR-ENTRY(1) TO WS-I-INHIBIT
         MOVE MQI-IN-ATTR-ENTRY(2) TO WS-I-TYPE
         MOVE MQI-IN-ATTR-ENTRY(3) TO WS-I-MAXL
         MOVE WS-INQ-DETS TO WS-OK-MSG-9
```

```
      END-IF
*
*--MQCLOSE  QUEUE TO QM
      MOVE  'CLOSE'   TO   WS-FUNCTION.
      MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
      MOVE MQCC-OK    TO   WS-CCODE-VALUE.
      MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
      CALL 'MQCLOSE' USING  WS-HCONN-ADDR-AREA
                            WS-HOBJ-ADDR-AREA
                            WS-Q-OPEN-OPTIONS
                            WS-CCODE-ADDR-AREA
                            WS-RCODE-ADDR-AREA.
*
      IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        THEN
                GO TO  9900-ERR-DISPLAY.

*--MQDISC  FROM QM
      MOVE  'DISCONN' TO   WS-FUNCTION.
      MOVE MQCC-OK    TO   WS-CCODE-VALUE.
      MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
      CALL 'MQDISC'  USING
                            WS-HCONN-ADDR-AREA
                            WS-CCODE-ADDR-AREA
                            WS-RCODE-ADDR-AREA.
*
      IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 6500-EXIT.
     EXIT.
*-----------------------------------------------------------------*
 7000-SEND-TOTALS.
*-----------------------------------------------------------------*
*--GET DURACTION TIME
     EXEC CICS ASKTIME
               ABSTIME(WS-ABSTIME2)
          END-EXEC.

     SUBTRACT WS-ABSTIME  FROM WS-ABSTIME2.
     EXEC CICS FORMATTIME
               ABSTIME (WS-ABSTIME2)
               TIME  (WS-DURATION-SECS)
               TIMESEP(':')
          END-EXEC.
*

     MOVE  WS-COUNT          TO   WS-OK-MESSAGES.
     MOVE  WS-DURATION-SECS  TO   WS-OK-TIME.
     MOVE  WS-DATA-QUEUE     TO   WS-OK-QUEUE.
     IF  WS-PUT-WITH-REPLY
      THEN
         MOVE  WS-REPLY-Q        TO  WS-OK-QUEUE-REPLY
         MOVE WS-OK-STATS-LINE-2 TO  WS-OK-MSG-2.

*-- --MOVE REST
     MOVE WS-OK-STATS-LINE-1 TO  WS-OK-MSG-1.
     MOVE WS-OK-STATS-LINE-3 TO  WS-OK-MSG-3.
     MOVE WS-OK-STATS-LINE-4 TO  WS-OK-MSG-4.
*-- --CHECK IF ANY ERRORS
     IF WS-END-OF-MESSAGES
      THEN
            MOVE 'NO MORE MESSAGES'    TO WS-OK-MSG-5.
*
     IF WS-TRUNCATED-MESSAGES
```

```
      THEN
            MOVE 'TRUNCATED MESSAGES'  TO WS-OK-MSG-6.
*
     IF WS-ERR-MSG
        EXEC CICS SEND
                  FROM   (WS-ALL-MSG)
                  LENGTH (LENGTH OF WS-ALL-MSG)
                  ERASE
        END-EXEC
      ELSE
        EXEC CICS SEND
                  FROM   (WS-OK-MSG)
                  LENGTH (LENGTH OF WS-OK-MSG)
                  ERASE
        END-EXEC.

*---------------------------------------------------------------*
   EJECT
*---------------------------------------------------------------*
 8000-GET-TIME-STAMP.
*---------------------------------------------------------------*
     EXEC  CICS ASKTIME
                ABSTIME(WS-ABSTIME)
           END-EXEC.
*
     EXEC  CICS FORMATTIME
                ABSTIME(WS-ABSTIME)
                YYMMDD (WS-DATE-YYMMDD)
           END-EXEC.

*
     MOVE  EIBTIME   TO   WS-TIME-9.
*
     MOVE WS-DATE-YYMMDD TO   WS-TIMESTAMP-DATE.
     MOVE WS-TIME-HHMMSS TO   WS-TIMESTAMP-TIME.

*---------------------------------------------------------------*
   EJECT
*---------------------------------------------------------------*
 9900-ERR-DISPLAY.
*---------------------------------------------------------------*
*--ERROR IN "MQ" VERB
*
     SET WS-ERR-MSG  TO TRUE.
     MOVE WS-CCODE-VALUE TO  WS-ERR-DISPLAY-CCODE.
     MOVE WS-RCODE-VALUE TO  WS-ERR-DISPLAY-RCODE.
*
     MOVE WS-ERR-DISPLAY TO  WS-ERR-LINES.
*
     GO TO  0000-SEND-TOTALS.
*
     GO TO  0000-RETURN.
```

Sample program TTPTST3.Z

This program is a test facility for putting and getting messages, by starting the sample transaction TST2 (program ID TTPTST2). It can be invoked:

- By passing data, using a CICS “START” command.
- By the terminal input “TST3”, which produces the screen requesting more input shown in Figure 63 on page 397.

```
07/10/2002          IBM MQSeries for VSE/ESA Version 2.1.2          MQBDTS
15:50:37             Test System Programs 3 - STARTS                CIC1
                                                                    A001

                    Async TASK Information
        Number of tasks....:


                    Message Processing Information
        Number of messages.:
        Function...........:          P=PUT, or G=GET

        PUT QUEUE name.....:
        PUT message size...:
        PUT message .......:
        PUT TimeStamp......:       Y=Yes, N=No




ENTER START VALUES.

    ENTER = Process              PF3 = Quit
```

Figure 63. Test System Programs 3 - start

On this screen the fields are:

Number of tasks	The number of asynchronous tasks (TST2 transactions).
Number of messages	The number of messages to be sent and received.
Function	Specify “P” to put message, “G” to get message.
PUT queue name	The queue name to be PUT or GET.
PUT message size	For the PUT function only, to specify the size of the message. If the PUT timestamp option is selected, the message is 16 characters larger than the PUT message size.
PUT message	The content of the message.
PUT TimeStamp	For the PUT function only, to put a time stamp in the message in the format YYMMDDHHMMSS. If you specify this option, the actual message size is 16 characters larger than the size specified in the PUT message size.

TTPTST3.Z

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    TTPTST3.
 DATE-COMPILED.
*-------------------------------------------------------------*
*
* SUMMARY CHANGES:                  TRACE POINT CODE: 109
*-------------------------------------------------------------
*
* CHANGE  DATE       PGM   COMMENT
*
*
*                                                             *
*-------------------------------------------------------------*
*                                                             *
*  TEST PROGRAMS TO START ASYNC TASKS                         *
```

TTPTST3.Z

```
*                                                                *
*              IBM MQI SYSTEM                                    *
*                                                                *
*----------------------------------------------------------------*
* PURPOSE: START ASYNC TASK     DEFINITION                       *
*----------------------------------------------------------------*
*                                                                *
* COPYBOOKS:  TTMTST3  - COBOL MAP SYMBOLIC                      *
*             MQIMTP   - MASTER TERMINAL COMMAREA                *
*             TTITST2  - TTPTST2 COMMAREA FOR STARTS             *
*             MQIERRWS - ERROR VALUES                            *
*             MQIERRCD - ERROR CODE                              *
*             TTETST3  - ERROR MESSAGES                          *
*             MQIENV   - ENVIRONMENT                             *
*             DFHAID   - 3270 AID DEFINITION                     *
*             DFHBMSCA - 3270 BMS CONTROL CHARACTERS             *
*                                                                *
* TRANSACTION:   TST3  - MASTER TERMINAL  (UPDATE )              *
*                                                                *
* MAPSET:        TTMTST3                                         *
* MAPS:          MAIN    - MAIN                                  *
*                                                                *
* SUMMARY CHANGES:                                               *
*                                                                *
*----------------------------------------------------------------*
     EJECT
*----------------------------------------------------------------*
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*----------------------------------------------------------------*
*    COPY COPYRWS.
*----------------------------------------------------------------*
* COPYRIGHT WORKING STORAGE  FOR COBOL MODULES                    *
*----------------------------------------------------------------*
 01   FILLER.
      05 FILLER                  PIC X(72) VALUE
      'Licensed Materials - Property of IBM'.
      05 FILLER                  PIC X(72) VALUE SPACES.
      05 FILLER                  PIC X(72) VALUE
      '5686-A06  '.
      05 FILLER                  PIC X(72) VALUE SPACES.
      05 FILLER                  PIC X(72) VALUE
      '(C) Copyright IBM Corp. 1998, 2002       All Rights Reserve
-     'd'.
      05 FILLER                  PIC X(72) VALUE SPACES.
      05 FILLER                  PIC X(72) VALUE
      'US Government Users Restricted Rights - Use, duplication or
-     '  '.
      05 FILLER                  PIC X(72) VALUE
      'disclosure restricted by GSA ADP Schedule Contract with IBM
-     ' Corp.'.
*----------------------------------------------------------------*
*Debugging eyecatcher information for start of WORKING-STORAGE.
*----------------------------------------------------------------*
 01  WS-RWS-PROGRAM-NAME1        PIC X(8).
 01  FILLER                      PIC X(16) VALUE
      ' Version V2.1.0'.
 01  WS-RWS-WHEN-COMPILED        PIC X(21).
 01  FILLER                      PIC X(7)  VALUE '========'.
*----------------------------------------------------------------*


 01  WS-VALUES.
     05  WS-CONFIGURATION-ADDRESS USAGE IS POINTER VALUE NULL.
     05  WS-REC-SIZE             PIC S9(4) COMP VALUE ZERO.
     05  WS-SS-STARTS            PIC  9(4)      VALUE ZERO.
```

```
     05  WS-NUM                 PIC  9(8)      VALUE ZERO.
     05  WS-NUM4                PIC  9(4)      VALUE ZERO.

     05  WS-APPLID              PIC X(8)   VALUE SPACES.
     05  WS-SYSID               PIC X(4)   VALUE SPACES.
     05  WS-STARTCD             PIC XX     VALUE SPACES.
       88 WS-STARTED                       VALUE 'SD'.

     05  WS-ABSTIME               PIC S9(15) COMP-3.
     05  WS-DATE-CCYYMMDD.
        10  WS-DATE-CC            PIC  99  VALUE ZERO.
        10  WS-DATE-YYMMDD.
          12  WS-DATE-YY          PIC  99  VALUE ZERO.
          12  WS-DATE-MM          PIC  99  VALUE ZERO.
          12  WS-DATE-DD          PIC  99  VALUE ZERO.
          12  FILLER              PIC  XX  VALUE ZERO.

     05  WS-UNPACK-TIME-9          PIC 9(07) VALUE ZEROES.
     05  WS-UNPACK-TIME-X REDEFINES WS-UNPACK-TIME-9.
         10  FILLER               PIC  X(01).
         10  WS-TIME-HHMMSS.
            12  WS-TIME-HH        PIC  X(02).
            12  WS-TIME-MM        PIC  X(02).
            12  WS-TIME-SS        PIC  X(02).
     05  WS-FORMATTED-TIME.
         10  WS-FORMAT-TIME-HH    PIC  X(02) VALUE SPACES.
         10  FILLER               PIC  X(01) VALUE ':'.
         10  WS-FORMAT-TIME-MM    PIC  X(02) VALUE SPACES.
         10  FILLER               PIC  X(01) VALUE ':'.
         10  WS-FORMAT-TIME-SS    PIC  X(02) VALUE SPACES.
     05  WS-FORMATTED-DATE.
         10  WS-FORMAT-DATE-MM    PIC  X(02) VALUE SPACES.
         10  FILLER               PIC  X(01) VALUE '/'.
         10  WS-FORMAT-DATE-DD    PIC  X(02) VALUE SPACES.
         10  FILLER               PIC  X(01) VALUE '/'.
         10  WS-FORMAT-DATE-CC    PIC  X(02) VALUE SPACES.
         10  WS-FORMAT-DATE-YY    PIC  X(02) VALUE SPACES.
     05  WS-TRAN-ID               PIC  X(4)  VALUE SPACES.
     05  WS-EDIT-ERR-FLAG         PIC  X(1) VALUE 'N'.
      88 WS-EDIT-ERR                        VALUE 'Y'.

     05  WS-RECORD-FLAG           PIC  X      VALUE SPACES.
      88 WS-RECORD-FOUND          VALUE 'T'.
      88 WS-RECORD-NOT-FOUND      VALUE 'F'.

     05  WS-ERROR-MESSAGE         PIC  X(79)  VALUE SPACES.
     05  WS-ERR-COUNT             PIC S9(4) COMP VALUE ZEROS.
     05  WS-ERR-MAX               PIC S9(4) COMP VALUE +20.
     05  WS-ERR-MESSAGE  VALUE SPACES.
       10  WS-ERR-MSG             PIC  X(79) OCCURS 20 TIMES.

*---------------------------------------------------------------*
   EJECT
*---------------------------------------------------------------*
     COPY DFHAID.
   EJECT
*---------------------------------------------------------------*
     COPY DFHBMSCA.

   EJECT
*---------------------------------------------------------------*
* BMS MAP
*---------------------------------------------------------------*
     COPY TTMTST3.
*---------------------------------------------------------------*
* TST2 COMMAREA
*---------------------------------------------------------------*
```

TTPTST3.Z

```
 01  WS-TST2-COMMAREA.
*    COPY TTITST2.
*----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: TTITST2   ***          - BEGIN - *
*----------------------------------------------------------------*
*   3/ 4/93   REV:                                               *
*----------------------------------------------------------------*
*  MQPINIT1 COMMAREA                                             *
*----------------------------------------------------------------*

     05  TST2-PASSED-INFO.
         10 TST2-FUNCTION            PIC  X(4) VALUE 'PUT'.
           88 TST2-FUNCT-PUT                   VALUE 'PUT'.
           88 TST2-FUNCT-GET                   VALUE 'GET'.

         10 TST2-PUT-NUM-MSG         PIC S9(4) COMP VALUE ZERO.
         10 TST2-PUT-QUEUE-NAME      PIC  X(48) VALUE SPACES.
         10 TST2-PUT-MSG-SIZE        PIC S9(4) COMP VALUE ZERO.
         10 TST2-PUT-MSG             PIC  X(48) VALUE SPACES.
         10 TST2-PUT-MSG-TIMESTAMP   PIC  X     VALUE SPACES.
          88 TST2-PUT-MSG-W-TIMESTAMP           VALUE 'Y'.

*----------------------------------------------------------------*
* - END -        ***  COPYBOOK: TTITST2   ***            - END - *
*----------------------------------------------------------------*
     EJECT
*----------------------------------------------------------------*
* ENVIRONMENT VALUES
*----------------------------------------------------------------*
 01  FILLER.
*    COPY MQICENV.
*----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQICENV   ***          - BEGIN - *
*----------------------------------------------------------------*
*   ENVIRONMENT VALUE            - SYSTEM   (ENV)                *
*----------------------------------------------------------------*

     02 ENV-DEFINITION.
     03 ENV-DATA-FOR-SYSTEM.
        05 ENV-PRODUCT-INSTALLED       PIC  X(4) VALUE 'MQI '.
          88  ENV-PRODUCT-MQM         VALUE 'MQM '.

        05 ENV-PRODUCT-RUNTIME         PIC  X(4).
          88  ENV-PRODUCT-RT-MQM      VALUE 'MQM '.

        05 ENV-LANG-INFO.
           10 ENV-LANGUAGE-FILE-CODE   PIC  99 VALUE 01.
           10 ENV-LANGUAGE             PIC  X(24)
                                       VALUE 'ENGLISH'.
        05 ENV-DATE-FORMAT             PIC  99 VALUE 01.
          88 ENV-DATE-MMDDYY                  VALUE 01.
          88 ENV-DATE-YYMMDD                  VALUE 02.
          88 ENV-DATE-YYDDMM                  VALUE 03.
          88 ENV-DATE-YYDDD                   VALUE 04.
          88 ENV-DATE-DDMMYY                  VALUE 05.


     03 ENV-DATA-FOR-TRAN.

        05 ENV-MASTER-TERMINAL-TRAN.
           10 ENV-MT-MASTER-TASK-ID    PIC  X(4) VALUE 'MQMT'.
           10 ENV-MT-CONFIG-TASK-ID    PIC  X(4) VALUE 'MQMC'.
           10 ENV-MT-MONITOR-TASK-ID   PIC  X(4) VALUE 'MQMM'.
           10 ENV-MT-OPER-TASK-ID      PIC  X(4) VALUE 'MQMO'.
           10 ENV-MT-DISP-TASK-ID      PIC  X(4) VALUE 'MQBQ'.
           10 ENV-MT-QUEUE-TASK-ID     PIC  X(4) VALUE 'MQMQ'.
           10 ENV-MT-QUEUEI-TASK-ID    PIC  X(4) VALUE 'MQDQ'.
```

```
           10 ENV-MT-COM-TASK-ID       PIC  X(4) VALUE 'MQMH'.
           10 ENV-MT-COMI-TASK-ID      PIC  X(4) VALUE 'MQDH'.
           10 ENV-MT-SYS-TASK-ID       PIC  X(4) VALUE 'MQMS'.
           10 ENV-MT-SYSI-TASK-ID      PIC  X(4) VALUE 'MQDS'.
           10 ENV-MT-MONQ-TASK-ID      PIC  X(4) VALUE 'MQQM'.
           10 ENV-MT-MONC-TASK-ID      PIC  X(4) VALUE 'MQCM'.
           10 ENV-MT-SS-TASK-ID        PIC  X(4) VALUE 'MQMA'.
           10 ENV-MT-SC-TASK-ID        PIC  X(4) VALUE 'MQMB'.
           10 ENV-MT-SI-TASK-ID        PIC  X(4) VALUE 'MQMI'.
           10 ENV-MT-SR-TASK-ID        PIC  X(4) VALUE 'MQMR'.
           10 ENV-MT-SD-TASK-ID        PIC  X(4) VALUE 'MQMD'.
           10 FILLER                   PIC  X(4) VALUE SPACES.
           10 FILLER                   PIC  X(4) VALUE SPACES.
           10 FILLER                   PIC  X(4) VALUE SPACES.

        05 ENV-INTERNAL-ITEMS-TRAN.
           10 ENV-II-MONITOR           PIC  X(4) VALUE 'MQSM'.
           10 ENV-II-M-RECOVERY        PIC  X(4) VALUE 'MQSR'.
           10 ENV-II-Q-RECOVERY        PIC  X(4) VALUE 'MQSQ'.
           10 ENV-II-START-STOP        PIC  X(4) VALUE 'MQSS'.
           10 ENV-II-TRAN-AIP2         PIC  X(4) VALUE 'MQ02'.
           10 ENV-II-TRAN-COM-CHECKP   PIC  X(4) VALUE 'MQCP'.
           10 ENV-II-TRAN-QUE-DELETE   PIC  X(4) VALUE 'MQQD'.
           10 ENV-II-TRAN-QUE-DEL-ALL  PIC  X(4) VALUE 'MQQA'.
           10 ENV-II-TRAN-REORG-QUE    PIC  X(4) VALUE 'MQRG'.
           10 ENV-II-TRAN-TCPIP        PIC  X(4) VALUE 'MQTL'.
           10 FILLER                   PIC  X(4) VALUE SPACES.


     03 ENV-DATA-FOR-PROGRAMS.

        05 ENV-MASTER-TERMINAL-PROGRAMS.
           10 ENV-MT-MASTER-PROGRAM    PIC  X(8) VALUE 'MQPMTP'.
           10 ENV-MT-CONFIG-PROGRAM    PIC  X(8) VALUE 'MQPMCFG'.
           10 ENV-MT-MONITOR-PROGRAM   PIC  X(8) VALUE 'MQPMMON'.
           10 ENV-MT-OPER-PROGRAM      PIC  X(8) VALUE 'MQPMOPR'.
           10 ENV-MT-DISP-PROGRAM      PIC  X(8) VALUE 'MQPDISP'.
           10 ENV-MT-QUEUE-PROGRAM     PIC  X(8) VALUE 'MQPMQUE'.
           10 ENV-MT-QUEUEI-PROGRAM    PIC  X(8) VALUE 'MQPMQUE'.
           10 ENV-MT-COM-PROGRAM       PIC  X(8) VALUE 'MQPMCOM'.
           10 ENV-MT-COMI-PROGRAM      PIC  X(8) VALUE 'MQPMCOM'.
           10 ENV-MT-SYS-PROGRAM       PIC  X(8) VALUE 'MQPMSYS'.
           10 ENV-MT-SYSI-PROGRAM      PIC  X(8) VALUE 'MQPMSYS'.
           10 ENV-MT-MONQ-PROGRAM      PIC  X(8) VALUE 'MQPMMOQ'.
           10 ENV-MT-MONC-PROGRAM      PIC  X(8) VALUE 'MQPMMOC'.
           10 ENV-MT-SS-PROGRAM        PIC  X(8) VALUE 'MQPMSS'.
           10 ENV-MT-SC-PROGRAM        PIC  X(8) VALUE 'MQPMSC'.
           10 ENV-MT-SI-PROGRAM        PIC  X(8) VALUE 'MQPMSI'.
           10 ENV-MT-SR-PROGRAM        PIC  X(8) VALUE 'MQPMMSN'.
           10 ENV-MT-SD-PROGRAM        PIC  X(8) VALUE 'MQPMDEL'.
           10 ENV-MT-CMD-PROGRAM       PIC  X(8) VALUE 'MQPCMD'.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.

        05 ENV-INTERNAL-ITEMS-PROGRAMS.
           10 ENV-II-LINK-ERROR        PIC  X(8) VALUE 'MQPERR  '.
           10 ENV-II-LINK-EIB1         PIC  X(8) VALUE 'MQPEIB1 '.
           10 ENV-II-LINK-AIP0         PIC  X(8) VALUE 'MQPAIP0 '.
           10 ENV-II-LINK-AIP1         PIC  X(8) VALUE 'MQPAIP1 '.
           10 ENV-II-LINK-AIP2         PIC  X(8) VALUE 'MQPAIP2 '.

           10 ENV-II-LINK-ECHO         PIC  X(8) VALUE 'MQPECHO '.
           10 ENV-II-LINK-FINDQ        PIC  X(8) VALUE 'MQPFINDQ'.
           10 ENV-II-LINK-QUE1         PIC  X(8) VALUE 'MQPQUE1 '.
           10 ENV-II-LINK-QUE2         PIC  X(8) VALUE 'MQPQUE2 '.
           10 ENV-II-LINK-INIT1        PIC  X(8) VALUE 'MQPINIT1'.
           10 ENV-II-LINK-INIT2        PIC  X(8) VALUE 'MQPINIT2'.
```

```
           10 ENV-II-LINK-SSQ          PIC  X(8) VALUE 'MQPSSQ  '.
           10 ENV-II-LINK-SCHK         PIC  X(8) VALUE 'MQPSCHK '.
           10 ENV-II-LINK-SREC         PIC  X(8) VALUE 'MQPSREC '.
           10 ENV-II-LINK-QRECOVERY    PIC  X(8) VALUE 'MQPQREC '.
           10 ENV-II-LINK-SENDER       PIC  X(8) VALUE 'MQPSEND '.
           10 ENV-II-LINK-RECIEVER     PIC  X(8) VALUE 'MQPRECV '.
           10 ENV-II-LINK-COM-CHECKP   PIC  X(8) VALUE 'MQPCCKPT'.
           10 ENV-II-LINK-QUE-DELETE   PIC  X(8) VALUE 'MQPQDEL'.
           10 ENV-II-LINK-SET-MAP      PIC  X(8) VALUE 'MQPSMAP'.
           10 ENV-II-LINK-MQPVSAM      PIC  X(8) VALUE 'MQPVSAM'.
*-- This module is no longer used. As a dirty fix the area has
*-- been reused to hold the code page information.
*          10 ENV-II-LINK-LU21         PIC  X(8) VALUE 'MQPLU21'.
           10 ENV-CODE-PAGE            PIC S9(4) COMP.
           10 ENV-TCPIP-PORT-NUMBER    PIC 9(8)  COMP.
           10 FILLER                   PIC XX.

           10 ENV-II-LINK-REPORT       PIC  X(8) VALUE 'MQPRPRT'.
           10 ENV-II-LINK-MQPCMD       PIC  X(8) VALUE 'MQPCMD'.
           10 ENV-II-LINK-MQPCLNT      PIC  X(8) VALUE 'MQPEXT1'.

      03 ENV-DATA-FOR-MAPS.

         05 ENV-MASTER-TERMINAL-MAPS.
           10 ENV-MT-MASTER-MAPSCREEN  PIC  X(8) VALUE 'MQMMTP'.
           10 ENV-MT-CONFIG-MAPSCREEN  PIC  X(8) VALUE 'MQMMCFG'.
           10 ENV-MT-MONITOR-MAPSCREEN PIC  X(8) VALUE 'MQMMMON'.
           10 ENV-MT-OPER-MAPSCREEN    PIC  X(8) VALUE 'MQMMOPR'.
           10 ENV-MT-DISP-MAPSCREEN    PIC  X(8) VALUE 'MQMDISP'.
           10 ENV-MT-QUEUE-MAPSCREEN   PIC  X(8) VALUE 'MQMMQUE'.
           10 ENV-MT-QUEUEI-MAPSCREEN  PIC  X(8) VALUE 'MQMMQUE'.
           10 ENV-MT-COM-MAPSCREEN     PIC  X(8) VALUE 'MQMMCOM'.
           10 ENV-MT-COMI-MAPSCREEN    PIC  X(8) VALUE 'MQMMCOM'.
           10 ENV-MT-SYS-MAPSCREEN     PIC  X(8) VALUE 'MQMMSYS'.
           10 ENV-MT-SYSI-MAPSCREEN    PIC  X(8) VALUE 'MQMMSYS'.
           10 ENV-MT-MONQ-MAPSCREEN    PIC  X(8) VALUE 'MQMMMOQ'.
           10 ENV-MT-MONC-MAPSCREEN    PIC  X(8) VALUE 'MQMMMOC'.
           10 ENV-MT-SS-MAPSCREEN      PIC  X(8) VALUE 'MQMMSS'.
           10 ENV-MT-SC-MAPSCREEN      PIC  X(8) VALUE 'MQMMSC'.
           10 ENV-MT-SI-MAPSCREEN      PIC  X(8) VALUE 'MQMMSI'.
           10 ENV-MT-SR-MAPSCREEN      PIC  X(8) VALUE 'MQMMMSN'.
           10 ENV-MT-SD-MAPSCREEN      PIC  X(8) VALUE 'MQMMDEL'.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.


      03 ENV-DATA-FOR-CONSTANTS.

         05 ENV-CONFIG-DDNAME          PIC  X(8) VALUE 'MQFCNFG'.
         05 ENV-SYSTEM-NUMBER          PIC  9(4) VALUE 1.
         05 ENV-MASTER-TERMINAL-CONS.
           10 ENV-MT-TITLE             PIC  X(40) VALUE
         '   IBM MQSeries for VSE/ESA Version 2 '.

         05 ENV-INTERNAL-ITEMS-CONS.
           10 ENV-II-ERROR-TD          PIC  X(4) VALUE 'MQER'.
           10 ENV-II-ERROR-CSMT        PIC  X(4) VALUE 'CSMT'.
           10 ENV-II-SYSTEM-ANCHOR     PIC  X(8) VALUE 'MQTAQM'.
           10 ENV-II-SYSTEM-PREFIX     PIC  X(4) VALUE 'MQI '.
           10 ENV-II-DUMPCODE          PIC  X(4) VALUE 'MQ??'.
           10 ENV-II-ENQ-INIT1         PIC  X(8) VALUE 'MQPINIT1'.
           10 ENV-II-SYSTEM-ENVIR      PIC  X(8) VALUE 'MQTENV  '.
           10 ENV-IT-UN-INIT-MSG       PIC  X(80) VALUE
        'MQI900000: MQSERIES VSE ENVIRONMENT NOT INITIALIZED.'.
           10 FILLER                   PIC  X(80) VALUE SPACES.
```

```
*-----------------------------------------------------------------*
* - END   -      ***  COPYBOOK: MQICENV   ***          - END   - *
*-----------------------------------------------------------------*
     EJECT
*-----------------------------------------------------------------*
* FINDQ COMMAREA
*-----------------------------------------------------------------*
 01  WS-FINDQ.
*    COPY MQIFINDQ.
*-----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQIFINDQ  ***          - BEGIN - *
*-----------------------------------------------------------------*
*   9/ 1/93   REV:                                                *
*-----------------------------------------------------------------*
*  FIND QUEUE    CALL PARAMETERS.                                 *
*-----------------------------------------------------------------*
     02  FINDQ-CALL-PARAMETERS.

*--PASSED INFO...
     03  FINDQ-PASSED-PARAMETERS.
         05  FINDQ-CALL-TYPE         PIC  X     VALUE SPACES.
             88  FINDQ-QUEUE-LOOKUP         VALUE 'Q'.
             88  FINDQ-SYSTEM-STATUS-ONLY   VALUE 'S'.

         05  FILLER                  PIC  X     VALUE SPACES.
         05  FINDQ-CALL-SYSTEM-NUM   PIC  99    VALUE ZERO.

*-- --QUEUE INFO
         05  FINDQ-QM-QUEUE-NAME.
           10 FINDQ-QM-NAME          PIC  X(48) VALUE SPACES.
           10 FINDQ-QUEUE-NAME       PIC  X(48) VALUE SPACES.

*--RETURN INFO
*-- --SYSTEM RETURN (ALWAYS RETURNED)
     03  FINDQ-RETURNED-PARAMETERS.
         05  FINDQ-SYSTEM-CODE       PIC  X     VALUE SPACES.
             88  FINDQ-SYSTEM-ACTIVE        VALUE 'A'.
             88  FINDQ-SYSTEM-INACTIVE      VALUE 'I'.
             88  FINDQ-SYSTEM-UN-INIT       VALUE SPACE.

         05  FILLER                  PIC  XXX   VALUE SPACES.
*-- -- --SYSTEM INFO (NOT SET IF SYSTEM UN-INIT)
         05 FINDQ-DEFAULT-QM-INFO.
           10 FINDQ-DEFAULT-NAME              PIC  X(48).
           10 FINDQ-QM-DESCRIPTION            PIC  X(64).
           10 FINDQ-DEFAULT-MAX-MSG           PIC S9(8) COMP.
           10 FINDQ-DEFAULT-MAX-CONN          PIC S9(8) COMP.
           10 FINDQ-DEFAULT-MAX-HANDLES       PIC S9(8) COMP.
           10 FINDQ-DEFAULT-MAX-WAIT-MON      PIC S9(8) COMP.
           10 FINDQ-DEFAULT-MAX-WAIT-REC      PIC S9(8) COMP.
           10 FINDQ-DEFAULT-MAX-REC-TASKS     PIC S9(4) COMP.
           10 FILLER                          PIC  XX.
           10 FINDQ-CONFIG-FILE               PIC  X(8).
             88 FINDQ-CONFIG-FILE-OK     VALUE 'MQFCNFG'.

           10 FINDQ-DEADLETTER-NAME           PIC  X(48).
           10 FINDQ-LOG-NAME                  PIC  X(48).
           10 FINDQ-AUDIT-NAME                PIC  X(48).
           10 FINDQ-MONITOR-NAME              PIC  X(48).
           10 FINDQ-ERROR-NAME                PIC  X(48).
           10 FINDQ-MONITOR-SYS-FLAG          PIC  X.
             88 FINDQ-MONITOR-ON         VALUE 'Y'.

           10 FINDQ-ERROR-TO-CSMT-FLAG        PIC  X.
             88 FINDQ-ERROR-TO-CSMT      VALUE 'Y', 'B'.
             88 FINDQ-ERROR-TO-BOTH      VALUE 'B'.
```

```
           10 FILLER                          PIC  XX.

      *-- --QUEUE  RETURN (ONLY RETURNED IF QUEUE REQUESTED)
               05  FINDQ-QUEUE-CODE        PIC  X     VALUE SPACES.
                   88  FINDQ-QUEUE-OK             VALUE  'Y'.
                   88  FINDQ-QUEUE-NOT-FOUND      VALUE SPACES.

               05  FILLER                  PIC  XXX   VALUE SPACES.

      *-- -- --ACTUAL MQI RETURN CODE
               05  FINDQ-QUEUE-ERROR-CODE  PIC S9(8) COMP VALUE ZERO.

      *-- -- --QUEUE INFO (NOT RETURNED IF QUEUE NOT-FOUND)
               05  FINDQ-RESOLVED-QM-QUEUE-NAME.
                 10 FINDQ-R-QM-NAME        PIC  X(48) VALUE SPACES.
                 10 FINDQ-R-QUEUE-NAME     PIC  X(48) VALUE SPACES.

               05  FINDQ-RESOLVED-LOCAL-NAME
                                           PIC X(48) VALUE SPACES.
               05  FINDQ-QUEUE-DRQ-ITEM    PIC S9(4) COMP VALUE ZERO.
               05  FILLER                  PIC  XX    VALUE SPACES.

      *-- -- -- --STATUS FROM DRQ
               05  FINDQ-RESOLVE-STATUS.
                 10 FINDQ-R-INBOUND-STAT   PIC  XX    VALUE SPACES.
                 10 FINDQ-R-OUTBOUND-STAT  PIC  XX    VALUE SPACES.

      *-- -- -- --ORIGINAL QUEUE VALUES
               05 FINDQ-QUEUE-DATA.
                 10 FINDQ-ADDED-DATA.
                    15 FINDQ-ADDED-TIME             PIC  X(6).
                    15 FILLER                       PIC  XX.
                    15 FINDQ-ADDED-DATE             PIC  X(8).
                    15 FINDQ-ADDED-TERMID           PIC  X(8).
                    15 FINDQ-ADDED-USERID           PIC  X(3).
                    15 FILLER                       PIC  X.

                 10 FINDQ-DESCRIPTION               PIC  X(64).
                 10 FINDQ-TYPE                      PIC X.
                   88 FINDQ-QUEUE-DEFINITION                VALUE
                                      'L', 'X', 'A', 'R', 'M'.
                   88 FINDQ-LOCAL-Q-ENTRY                   VALUE 'L'.
                   88 FINDQ-LOCAL-AIX-Q                     VALUE 'X'.
                   88 FINDQ-ALIAS-Q-ENTRY                   VALUE 'A'.
                   88 FINDQ-REMOTE-Q-ENTRY                  VALUE 'R'.
                   88 FINDQ-MODEL-Q-ENTRY                   VALUE 'M'.

                 10 FINDQ-TYPE-ALIAS                PIC X.
                   88 FINDQ-ALIAS-QUEUE                     VALUE 'Q'.
                   88 FINDQ-ALIAS-MANAGER                   VALUE 'M'.
                   88 FINDQ-ALIAS-REPLY                     VALUE 'R'.

                 10 FILLER                          PIC XX.

                 10 FINDQ-ATTR-FLAGS.
                    15 FINDQ-INHIBIT-PUT-FLAG       PIC X.
                     88 FINDQ-INHIBIT-PUT                    VALUE 'Y'.
                    15 FINDQ-INHIBIT-GET-FLAG       PIC X.
                     88 FINDQ-INHIBIT-GET                    VALUE 'Y'.

                    15 FINDQ-PERSIST-FLAG           PIC X.
                     88 FINDQ-PERSIST-DEFAULT                VALUE 'Y'.

                 10 FILLER                          PIC X.

               05 FINDQ-LOCAL-INFO.
                 10 FINDQ-DEFINITION-FLAG     PIC  X.
```

```
             88 FINDQ-DEF-PERM           VALUE 'Y'.
             88 FINDQ-DEF-NOT-PERM       VALUE 'N'.

           10 FINDQ-USAGE-MODE-FLAG     PIC  X.
             88 FINDQ-U-MODE-NORMAL      VALUE 'N'.
             88 FINDQ-U-MODE-TRANSM      VALUE 'Y'.

           10 FINDQ-SHAREABLE-FLAG      PIC  X.
             88 FINDQ-SHARE-QUEUE        VALUE 'Y'.
             88 FINDQ-NON-SHARE-QUEUE    VALUE 'N'.

           10 FINDQ-TRIGGER-TYPE        PIC  X.
            88 FINDQ-NO-TRIGGER          VALUE SPACE.
            88 FINDQ-TRIGGER-ON          VALUE 'Y'.

*-----------------------------------------------------------------*
* - END -        ***  COPYBOOK: MQIFINDQ  ***            - END - *
*-----------------------------------------------------------------*

     EJECT
*-----------------------------------------------------------------*
* COMMAREA
*-----------------------------------------------------------------*
*    COPY MQIMTP.
*-----------------------------------------------------------------*
*-----------------------------------------------------------------*
* COPYBOOK: MQIMTP                                                *
*                                                                 *
* FUNCTION: COMMAREA FOR MASTER TERMINAL TASK                     *
*                                                                 *
*-----------------------------------------------------------------*
*-----------------------------------------------------------------*
 01  MTP-COMMAREA.
     05  MTP-HEADER-FLAG     PIC X(4) VALUE 'MQI '.
         88 MTP-HEADER-OK             VALUE 'MQI '.

     05  MTP-MAIN-TASK       PIC X(4) VALUE SPACES.
       88 MTP-NO-RETURN-TASK          VALUE SPACES.

     05  MTP-ACTIVE-TASK     PIC X(4) VALUE SPACES.

     05  MTP-MAP-VALUE       PIC X(8) VALUE 'MAIN'.
         88 MTP-MAP-MAIN            VALUE 'MAIN'.
         88 MTP-MAP-OPTIONS         VALUE 'OPTIONS'.
         88 MTP-MAP-QUEUE           VALUE 'QUEUE  '.
         88 MTP-MAP-LOCAL           VALUE 'LOCAL  '.
         88 MTP-MAP-QLIST           VALUE 'QLIST  '.
         88 MTP-MAP-REORG           VALUE 'REORG  '.

     05  MTP-SCREEN-IND      PIC X    VALUE SPACE.
         88 MTP-SCREEN-FIRST        VALUE 'F'.
         88 MTP-SCREEN-RETURN       VALUE SPACE.
         88 MTP-SCREEN-SEND         VALUE 'S'.
         88 MTP-SCREEN-RECEIVE      VALUE 'R'.

     05  MTP-MAP-FUNCTION    PIC X(8) VALUE 'DISPLAY'.
         88 MTP-MAP-DISPLAY         VALUE 'DISPLAY'.
         88 MTP-MAP-LIST            VALUE 'LIST'.
         88 MTP-MAP-ADD             VALUE 'ADD    '.
         88 MTP-MAP-UPDATE          VALUE 'UPDATE '.
         88 MTP-MAP-DELETE          VALUE 'DELETE '.

     05  MTP-CONFIG-FILE     PIC X(8)  VALUE SPACE.
     05  MTP-SYSTEM-REC-FLAG PIC X  VALUE SPACE.
         88 MTP-SYSTEM-REC-FOUND    VALUE 'Y'.
         88 MTP-SYSTEM-REC-NOTFOUND VALUE 'N'.
```

```
     05  MTP-CONFIRM-IND     PIC X VALUE SPACE.
         88  MTP-CONFIRM           VALUE 'Y'.
         88  MTP-NO-CONFIRM        VALUE 'N'.

     05  FILLER              PIC XX     VALUE SPACE.

*--CONFIGURATION DATA
     05  MTP-CONFIG-DATA     PIC X(4) VALUE SPACES.

*--GENERAT EXTENDED DATA
     05  MTP-EXTENDED-COMMAREA.
         10 FILLER           PIC X(2000) VALUE SPACES.

*----------------------------------------------------------------*
*----------------------------------------------------------------*
     EJECT
*----------------------------------------------------------------*
* CONFIGURATION FILE
*----------------------------------------------------------------*
*    COPY MQICONFG.
*----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQICONFG  ***          - BEGIN - *
*----------------------------------------------------------------*
*----------------------------------------------------------------*
*    CONFIGURATION FILE                                          *
*----------------------------------------------------------------*

*----------------------------------------------------------------*
*   MAIN CONFIGURATION RECORD                                    *
*----------------------------------------------------------------*
 01  CONFIGURATION-RECORD VALUE SPACES.
     03 FILLER                           PIC  X(2048).



*----------------------------------------------------------------*
*   ENVIRONMENT VALUE            - SYSTEM   (ENV)                *
*----------------------------------------------------------------*

 01  ENVIRONMENT-RECORD
     REDEFINES  CONFIGURATION-RECORD.
     03 ENV-RECORD-KEY.
        05 ENV-RECORD-ID                 PIC  X(4).
          88 RECORD-TYPE-IS-ENV               VALUE 'ENV'.
        05 ENV-RECORD-VERSION            PIC  9(4).
        05 ENV-RECORD-TYPE               PIC  X(4).
          88 ENV-TYPE-SYSTEM                  VALUE 'SYS'.
          88 ENV-TYPE-TRANACTION              VALUE 'TRAN'.
          88 ENV-TYPE-PROGRAM                 VALUE 'PROG'.
          88 ENV-TYPE-MAPS                    VALUE 'MAPS'.
          88 ENV-TYPE-CONSTANTS               VALUE 'CONS'.
        05 ENV-FILLER                    PIC  X(88).

     03 ENV-LAST-MAINTAINED-DATA.
        05 ENV-LAST-TIME                 PIC  9(6).
        05 FILLER                        PIC  XX.
        05 ENV-LAST-DATE                 PIC  X(8).
        05 ENV-LAST-TERMID               PIC  X(4).
        05 ENV-LAST-USERID               PIC  X(8).

     03 ENV-ADDED-MAINTAINED-DATA.
        05 ENV-ADDED-TIME                PIC  9(6).
        05 FILLER                        PIC  XX.
        05 ENV-ADDED-DATE                PIC  X(8).
        05 ENV-ADDED-TERMID              PIC  X(4).
        05 ENV-ADDED-USERID              PIC  X(8).
```

```
     03 ENV-DATA-AREA.
        05 FILLER                          PIC  X(1892).

*---------------------------------------------------------------*
*   SYSTEM DESCRIPTOR RECORD (SYS)                              *
*---------------------------------------------------------------*

 01  SYSTEM-DESCRIPTOR-RECORD
     REDEFINES  CONFIGURATION-RECORD.
     03 SYS-RECORD-KEY.
        05 SYS-RECORD-ID                   PIC  X(4).
          88 RECORD-TYPE-IS-SYS               VALUE 'SYS'.
        05 SYS-RECORD-SYSTEM-NUMBER        PIC  9(4).
        05 SYS-RECORD-TYPE                 PIC  X(4).
          88 SYS-TYPE-SYS                     VALUE 'SYS'.
          88 SYS-TYPE-QUE-MAX                 VALUE 'QUEM'.
          88 SYS-TYPE-QUE-DEFAULT             VALUE 'QUED'.
          88 SYS-TYPE-COM-MAX                 VALUE 'COMM'.
          88 SYS-TYPE-COM-DEFAULT             VALUE 'COMD'.
          88 SYS-TYPE-COM-PARM                VALUE 'COMP'.
        05 SYS-FILLER                      PIC  X(88).

     03 SYS-LAST-MAINTAINED-DATA.
        05 SYS-LAST-TIME                   PIC  9(6).
        05 FILLER                          PIC  XX.
        05 SYS-LAST-DATE                   PIC  X(8).
        05 SYS-LAST-TERMID                 PIC  X(4).
        05 SYS-LAST-USERID                 PIC  X(8).

     03 SYS-ADDED-MAINTAINED-DATA.
        05 SYS-ADDED-TIME                  PIC  9(6).
        05 FILLER                          PIC  XX.
        05 SYS-ADDED-DATE                  PIC  X(8).
        05 SYS-ADDED-TERMID                PIC  X(4).
        05 SYS-ADDED-USERID                PIC  X(8).

     03 SYS-DATA.
        05 FILLER                          PIC  X(1892).


*---------------------------------------------------------------*
*   QUEUE DESCRIPTOR RECORD (QDR)                               *
*---------------------------------------------------------------*

 01  QUEUE-DESCRIPTOR-RECORD
     REDEFINES  CONFIGURATION-RECORD.
     03 QDR-RECORD-KEY.
        05 QDR-RECORD-ID                   PIC  X(4).
          88 RECORD-TYPE-IS-QDR                    VALUE 'QDR'.
        05 QDR-RECORD-SYSTEM-NUMBER        PIC  9(4).
        05 QDR-OBJ-NAME                    PIC  X(48).
        05 FILLER                          PIC  X(44).

     03 QDR-LAST-MAINTAINED-DATA.
        05 QDR-LAST-TIME                   PIC  9(6).
        05 FILLER                          PIC  XX.
        05 QDR-LAST-DATE                   PIC  X(8).
        05 QDR-LAST-TERMID                 PIC  X(4).
        05 QDR-LAST-USERID                 PIC  X(8).

     03 QDR-ADDED-MAINTAINED-DATA.
        05 QDR-ADDED-TIME                  PIC  9(6).
        05 FILLER                          PIC  XX.
        05 QDR-ADDED-DATE                  PIC  X(8).
        05 QDR-ADDED-TERMID                PIC  X(4).
        05 QDR-ADDED-USERID                PIC  X(8).
```

```
     03 QDR-DATA.
        05 FILLER                          PIC  X(1892).


*---------------------------------------------------------------*
*   COMMUNICATION  (COM)                                        *
*---------------------------------------------------------------*

 01  COMMUNICATION-RECORD
     REDEFINES  CONFIGURATION-RECORD.
     03 COM-RECORD-KEY.
        05 COM-RECORD-ID                   PIC  X(4).
          88 RECORD-TYPE-IS-COM                    VALUE 'COM'.
        05 COM-RECORD-SYSTEM-NUMBER        PIC  9(4).
        05 COM-NAME                        PIC  X(20).
        05 COM-KEY-TYPE                    PIC  X.
        05 FILLER                          PIC  X(71).

     03 COM-LAST-MAINTAINED-DATA.
        05 COM-LAST-TIME                   PIC  9(6).
        05 FILLER                          PIC  XX.
        05 COM-LAST-DATE                   PIC  X(8).
        05 COM-LAST-TERMID                 PIC  X(4).
        05 COM-LAST-USERID                 PIC  X(8).

     03 COM-ADDED-MAINTAINED-DATA.
        05 COM-ADDED-TIME                  PIC  9(6).
        05 FILLER                          PIC  XX.
        05 COM-ADDED-DATE                  PIC  X(8).
        05 COM-ADDED-TERMID                PIC  X(4).
        05 COM-ADDED-USERID                PIC  X(8).


     03 COM-DATA.
        05 FILLER                          PIC  X(1892).


*---------------------------------------------------------------*
*   TEXTUAL RECORD (TEXT)                                       *
*---------------------------------------------------------------*

 01  TEXT-RECORD
     REDEFINES  CONFIGURATION-RECORD.
     03 TEXT-RECORD-KEY.
        05 TEXT-RECORD-ID                  PIC  X(4).
          88 RECORD-TYPE-IS-TEXT         VALUE 'TEXT'.

        05 TEXT-RECORD-SYSTEM-NUMBER       PIC  9(4).
        05 TEXT-RECORD-TYPE                PIC  X(4).
          88 TEXT-TYPE-MESSAGES          VALUE 'MSGS'.
          88 TEXT-TYPE-MAPS              VALUE 'MAPS'.
          88 TEXT-TYPE-HELP              VALUE 'HELP'.

        05 TEXT-HELP-KEY.
           10 TEXT-HELP-SCREEN             PIC  X(8).
           10 TEXT-HELP-FUNCTION           PIC  X(40).
           10 TEXT-HELP-FUNCT-SCREEN-NUM   PIC S9(4) COMP.


        05 TEXT-MAPS-KEY
             REDEFINES  TEXT-HELP-KEY.
           10 TEXT-MAPS-SCREEN             PIC  X(8).
             88 TEXT-MAPS-MAIN-TITLE       VALUE 'ALL'.

           10 TEXT-MAPS-MAPSET             PIC  X(8).
           10 TEXT-MAPS-MAPSET-TYPE        PIC  X(4).
            88 TEXT-MAPS-MAPSET-HEADER     VALUE 'HEAD'.
```

```
           88 TEXT-MAPS-MAPSET-DATA       VALUE 'DATA'.
           88 TEXT-MAPS-MAPSET-MSGS       VALUE 'MSGS'.

         10 TEXT-MAPS-SCREEN-DATA-NUM    PIC S9(4) COMP.


      05 TEXT-MESSAGE-KEY
           REDEFINES  TEXT-HELP-KEY.
         10 TEXT-MESS-NUMBER             PIC  9(6).
         10 TEXT-MESS-RECORD-NUM         PIC S9(4) COMP.

      05 FILLER                          PIC  X(38).

   03 TEXT-LAST-MAINTAINED-DATA.
      05 TEXT-LAST-TIME                  PIC  9(6).
      05 FILLER                          PIC  XX.
      05 TEXT-LAST-DATE                  PIC  X(8).
      05 TEXT-LAST-TERMID                PIC  X(4).
      05 TEXT-LAST-USERID                PIC  X(8).


   03 TEXT-ADDED-MAINTAINED-DATA.
      05 TEXT-ADDED-TIME                 PIC  9(6).
      05 FILLER                          PIC  XX.
      05 TEXT-ADDED-DATE                 PIC  X(8).
      05 TEXT-ADDED-TERMID               PIC  X(4).
      05 TEXT-ADDED-USERID               PIC  X(8).


   03 TEXT-DATA-AREA.
      05 FILLER                          PIC  X(1892).

*-----------------------------------------------------------------*
* - END -        ***  COPYBOOK: MQICONFG  ***           - END - *
*-----------------------------------------------------------------*



    EJECT
*-----------------------------------------------------------------*
* ERROR HANDLEING
*-----------------------------------------------------------------*
 01  WS-ERR.
*    COPY        MQIERR.
*-----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQIERR    ***         - BEGIN - *
*-----------------------------------------------------------------*
*      ERROR MODULE CALLING PARAMETERS                            *
*-----------------------------------------------------------------*

    02 ERR-HANDLER-COMMAREA.
       05 ERR-CURRENT-INFO.
         10 ERR-COM-HANDLER       PIC  X(48) VALUE SPACES.
         10 ERR-QUEUE             PIC  X(48) VALUE SPACES.
         10 ERR-FILE              PIC  X(8)  VALUE SPACES.
         10 ERR-DETAIL            PIC  X(80) VALUE SPACES.
         10 ERR-DETAIL2           PIC  X(80) VALUE SPACES.
         10 ERR-Q-CODE            PIC S9(8) COMP VALUE ZERO.
         10 FILLER                PIC  X(8) VALUE SPACES.

       05 ERR-RESULTS.
         10 ERR-CODE              PIC  9(6) VALUE ZERO.
         10 FILLER                PIC  XX   VALUE SPACES.
         10 ERR-PROGRAM           PIC  X(8) VALUE SPACES.
         10 ERR-TRANID            PIC  X(4) VALUE SPACES.
         10 ERR-TERMID            PIC  X(4) VALUE SPACES.
         10 ERR-TASKNO            PIC S9(7)  COMP-3 VALUE ZERO.
```

```
           10 ERR-ABSTIME          PIC S9(15) COMP-3 VALUE ZERO.

           10 ERR-DEBUG-EIBFN      PIC  XX   VALUE SPACES.
           10 ERR-DEBUG-EIBRCODE   PIC  X(6) VALUE LOW-VALUES.
           10 ERR-DEBUG-EIBRSRCE   PIC  X(8) VALUE LOW-VALUES.
           10 ERR-DEBUG-EIBRESP    PIC S9(8) COMP VALUE ZEROS.
           10 ERR-DEBUG-EIBRESP2   PIC S9(8) COMP VALUE ZEROS.
           10 ERR-DEBUG-EIBERRCD   PIC  X(4) VALUE LOW-VALUES.
           10 ERR-DEBUG-ABEND      PIC  X(4) VALUE SPACES.
           10 FILLER               PIC  X(12) VALUE SPACES.

      *---------------------------------------------------------------*
      * - END -        ***  COPYBOOK: MQIERR    ***           - END - *
      *---------------------------------------------------------------*

      *    COPY        MQIERRC.
      *---------------------------------------------------------------*
      * IBM MQSERIES COMMON ERROR CODES
      *---------------------------------------------------------------*
       01  MSG-ERROR-MESSAGES.
           05 ERR-NO-ENVIRONMENT          PIC  9(6) VALUE 900000.

           05 ERR-CICS-ERROR              PIC  9(6) VALUE 800000.
           05 ERR-CICS-INVALID-REQ        PIC  9(6) VALUE 800010.
           05 ERR-CICS-ILLOGIC            PIC  9(6) VALUE 800011.
           05 ERR-CICS-ERROR-CHECKPOINT   PIC  9(6) VALUE 800090.
           05 ERR-CICS-ABEND              PIC  9(6) VALUE 800099.
           05 ERR-CICS-FILE-NOTOPEN       PIC  9(6) VALUE 801012.
           05 ERR-CICS-DISABLE            PIC  9(6) VALUE 801019.
           05 ERR-CICS-NO-STORAGE         PIC  9(6) VALUE 802000.
           05 ERR-CICS-LENGTH-ERR         PIC  9(6) VALUE 803001.
           05 ERR-CICS-MAPFAIL            PIC  9(6) VALUE 808000.
           05 ERR-CICS-PGMIDERR           PIC  9(6) VALUE 809000.
           05 ERR-CICS-FILEID             PIC  9(6) VALUE 809010.
           05 ERR-CICS-NOFILE             PIC  9(6) VALUE 809011.
           05 ERR-CICS-IO-ERROR           PIC  9(6) VALUE 809012.
           05 ERR-CICS-TRANIDERR          PIC  9(6) VALUE 809050.

           05 ERR-COM-FREE-ERROR          PIC  9(6) VALUE 501001.
           05 ERR-COM-EIB-ERROR           PIC  9(6) VALUE 501002.
           05 ERR-COM-STAT-ERROR          PIC  9(6) VALUE 501003.
           05 ERR-COM-ALLOC-ERROR         PIC  9(6) VALUE 501004.
           05 ERR-COM-ALLOC-RETRY         PIC  9(6) VALUE 501005.
           05 ERR-COM-CONN-ERROR          PIC  9(6) VALUE 501006.
           05 ERR-COM-SEND-ERROR          PIC  9(6) VALUE 501008.
           05 ERR-COM-RECV-RESP-ERR       PIC  9(6) VALUE 501009.
           05 ERR-COM-RESP-TYPE           PIC  9(6) VALUE 501010.
           05 ERR-COM-RESP-MSN            PIC  9(6) VALUE 501011.
           05 ERR-COM-RESP-FATAL          PIC  9(6) VALUE 501012.
           05 ERR-COM-MSG-ERROR           PIC  9(6) VALUE 501013.
           05 ERR-COM-BIG-INDIAN          PIC  9(6) VALUE 501014.
           05 ERR-COM-TSH-ERROR           PIC  9(6) VALUE 501015.
           05 ERR-COM-CCSID-ERROR         PIC  9(6) VALUE 501016.
           05 ERR-COM-MSH-ERROR           PIC  9(6) VALUE 501017.
           05 ERR-COM-MQX-ERROR           PIC  9(6) VALUE 501018.
           05 ERR-COM-INIT-ERROR          PIC  9(6) VALUE 501019.
           05 ERR-COM-FAP-ERROR           PIC  9(6) VALUE 501020.
           05 ERR-COM-MSG-SIZE            PIC  9(6) VALUE 501021.
           05 ERR-COM-WRAP-ERROR          PIC  9(6) VALUE 501022.
           05 ERR-COM-MCP-DOWN            PIC  9(6) VALUE 501023.
           05 ERR-COM-DOWN                PIC  9(6) VALUE 501024.
           05 ERR-COM-NOT-FOUND           PIC  9(6) VALUE 501025.
           05 ERR-COM-ERROR               PIC  9(6) VALUE 501026.
           05 ERR-COM-BUSY                PIC  9(6) VALUE 501027.
           05 ERR-COM-RESYNC-ERROR        PIC  9(6) VALUE 501028.
           05 ERR-COM-STATUS-ERROR        PIC  9(6) VALUE 501029.
           05 ERR-COM-LENGTH-ERROR        PIC  9(6) VALUE 501030.
```

```
05 ERR-COM-MSG-PER-BATCH        PIC  9(6) VALUE 501031.
05 ERR-COM-MAX-TRANSM-SIZE      PIC  9(6) VALUE 501032.
05 ERR-COM-RESET-MSN            PIC  9(6) VALUE 501050.

05 ERR-INT-LINK-ERROR           PIC  9(6) VALUE 400000.
05 ERR-INT-LINK-COM-SIZE        PIC  9(6) VALUE 400001.
05 ERR-INT-LINK-COM-DATA        PIC  9(6) VALUE 400002.
05 ERR-INT-RETURN-ERROR         PIC  9(6) VALUE 400003.
05 ERR-INT-MOVE-ERROR           PIC  9(6) VALUE 400010.
05 ERR-INT-STRUC-MISSING        PIC  9(6) VALUE 402000.
05 ERR-INT-STRUC-ERROR          PIC  9(6) VALUE 402090.

05 ERR-LOGIC-NOT-SUPPORTED      PIC  9(6) VALUE 300000.
05 ERR-LOGIC-STARTED-WRONG      PIC  9(6) VALUE 300010.
05 ERR-LOGIC-REPEATED-FAILURE   PIC  9(6) VALUE 300020.
05 ERR-LOGIC-LOCKS-EXCEEDED     PIC  9(6) VALUE 300030.
05 ERR-LOGIC-MISSING-RECORD     PIC  9(6) VALUE 301000.
05 ERR-LOGIC-RECORD-DUPLICATED  PIC  9(6) VALUE 301010.
05 ERR-LOGIC-Q-CKP-MISSING      PIC  9(6) VALUE 309010.

05 ERR-PROC-SYSTEM-STOPPED      PIC  9(6) VALUE 100000.
05 ERR-PROC-SYSTEM-ACTIVE       PIC  9(6) VALUE 100010.
05 ERR-PROC-SYS-START-NOQDR     PIC  9(6) VALUE 100011.
05 ERR-PROC-SYS-START-MAXQDR    PIC  9(6) VALUE 100012.
05 ERR-PROC-SYS-START-MAXCOM    PIC  9(6) VALUE 100013.
05 ERR-PROC-SYS-START-NOSYS     PIC  9(6) VALUE 100090.
05 ERR-PROC-Q-EXCEEDED-DEPTH    PIC  9(6) VALUE 101000.
05 ERR-PROC-Q-CONCURRENT-UPD    PIC  9(6) VALUE 101010.
05 ERR-PROC-Q-NOTFOUND          PIC  9(6) VALUE 101015.
05 ERR-PROC-Q-STOPPED           PIC  9(6) VALUE 101090.
05 ERR-PROC-Q-DISABLED          PIC  9(6) VALUE 101091.
05 ERR-PROC-QSN-LIMIT-REACHED   PIC  9(6) VALUE 102090.
05 ERR-PROC-FILE-SPACE-PUT      PIC  9(6) VALUE 102091.
05 ERR-PROC-FILE-SPACE          PIC  9(6) VALUE 102092.
05 ERR-PROC-DUAL-Q-ERROR        PIC  9(6) VALUE 104021.
05 ERR-PROC-DUAL-Q-FILE         PIC  9(6) VALUE 104022.
05 ERR-PROC-DUAL-Q-LOGIC        PIC  9(6) VALUE 104023.
05 ERR-PROC-TRIGGER-ERROR       PIC  9(6) VALUE 105090.
05 ERR-PROC-TRIGGER-DATA        PIC  9(6) VALUE 105091.
05 ERR-PROC-NOT-AUTHORIZED      PIC  9(6) VALUE 109000.

05 ERR-WARN-SYS-STARTED-W-ERR   PIC  9(6) VALUE 010000.
05 ERR-WARN-SYS-STARTED-W-FILER PIC  9(6) VALUE 010001.
05 ERR-WARN-SYS-STARTED-W-COMER PIC  9(6) VALUE 010002.
05 ERR-WARN-SYS-STARTED-W-CHANG PIC  9(6) VALUE 010003.
05 ERR-WARN-SYS-QUEUE-DISABLED  PIC  9(6) VALUE 010005.
05 ERR-WARN-SYS-QUEUE-ENABLED   PIC  9(6) VALUE 010007.

05 ERR-WARN-COM-CONNECT         PIC  9(6) VALUE 005000.
05 ERR-WARN-COM-OPENED          PIC  9(6) VALUE 005001.
05 ERR-WARN-COM-QUEUE-OPENED    PIC  9(6) VALUE 005002.
05 ERR-WARN-COM-LU62-CONNECT    PIC  9(6) VALUE 005003.
05 ERR-WARN-COM-RECEIVER-ALLOC  PIC  9(6) VALUE 005004.
05 ERR-WARN-COM-QUEUE-EMPTY     PIC  9(6) VALUE 005005.
05 ERR-WARN-COM-QUEUE-CLOSED    PIC  9(6) VALUE 005006.
05 ERR-WARN-COM-DISC            PIC  9(6) VALUE 005007.
05 ERR-WARN-COM-SHUT            PIC  9(6) VALUE 005008.
05 ERR-WARN-COM-SHUT-SENT       PIC  9(6) VALUE 005009.

05 ERR-WARN-COM-TCP-CONNECT     PIC  9(6) VALUE 006003.
05 ERR-WARN-COM-TCP-STARTED     PIC  9(6) VALUE 006007.
05 TCP-FREE-ERROR               PIC  9(6) VALUE 006010.
05 TCP-CONN-ERROR               PIC  9(6) VALUE 006015.
05 TCP-SEND-ERROR               PIC  9(6) VALUE 006016.
05 TCP-RECV-RESP-ERROR          PIC  9(6) VALUE 006017.

05 ERR-FUNCTION-STARTED         PIC  9(6) VALUE 000100.
```

```
           05 ERR-FUNCTION-DONE            PIC  9(6) VALUE 001000.
           05 ERR-FUNCTION-NOT-DONE        PIC  9(6) VALUE 001090.

           05 ERR-WARN-SYS-STARTED         PIC  9(6) VALUE 000000.

           05  SYNCH-MSN-ERROR              PIC 9(6)    VALUE 3.
           05  SYNCH-MSG-DUP                PIC 9(6)    VALUE 4.
           05  ERR-WARN-COM-STARTED         PIC 9(6) VALUE 000007.
           05  ERR-WRAP-COM-MISMATCH        PIC 9(6) VALUE 000009.
           05  LU62-FREE-ERROR              PIC 9(6)    VALUE 10.
           05  LU62-EIB-ERROR               PIC 9(6)    VALUE 11.
           05  LU62-STAT-ERROR              PIC 9(6)    VALUE 12.
           05  LU62-ALLOC-ERROR             PIC 9(6)    VALUE 13.
           05  LU62-ALLOC-RETRY-ERROR       PIC 9(6)    VALUE 14.
           05  LU62-CONN-ERROR              PIC 9(6)    VALUE 15.
           05  LU62-SEND-ERROR              PIC 9(6)    VALUE 16.
           05  LU62-RECV-RESP-ERROR         PIC 9(6)    VALUE 17.
           05  INVLD-RESP-TYPE              PIC 9(6)    VALUE 23.
           05  INVLD-RESP-MSN               PIC 9(6)    VALUE 24.
           05  FATAL-RESP-TYPE              PIC 9(6)    VALUE 25.
           05  RECOVERABLE-RESP-TYPE        PIC 9(6)    VALUE 26.
           05  PARSER-MSN-ERROR             PIC 9(6)    VALUE 29.
           05  PARSER-TYPE-ERROR            PIC 9(6)    VALUE 30.
           05  PARSER-PDM-ERROR             PIC 9(6)    VALUE 31.
           05  PARSER-SID-ERROR             PIC 9(6)    VALUE 32.
           05  PARSER-PN-ERROR              PIC 9(6)    VALUE 33.
           05  PARSER-KEY-ERROR             PIC 9(6)    VALUE 34.
           05  PARSER-APID-ERROR            PIC 9(6)    VALUE 35.
           05  PARSER-ORG-DT-ERROR          PIC 9(6)    VALUE 38.
           05  PARSER-ORIG-MSN-ERROR        PIC 9(6)    VALUE 39.
           05  PARSER-BODY-ERROR            PIC 9(6)    VALUE 40.
           05  PARSER-STATUS-ERROR          PIC 9(6)    VALUE 41.
           05  PARSER-LENGTH-ERROR          PIC 9(6)    VALUE 42.
           05  MCCONN-ERROR                 PIC 9(6)    VALUE 51.
           05  MQOPEN-ERROR                 PIC 9(6)    VALUE 52.
           05  MQGET-ERROR                  PIC 9(6)    VALUE 53.
           05  MQPUT-ERROR                  PIC 9(6)    VALUE 54.
           05  MQPT1-ERROR                  PIC 9(6)    VALUE 55.
           05  MQCLOSE-ERROR                PIC 9(6)    VALUE 56.
           05  MQDISC-ERROR                 PIC 9(6)    VALUE 57.
           05  QM-OTHER-ERROR               PIC 9(6)    VALUE 60.
           05  RECV-RETURN-LON-STATUS       PIC 9(6)    VALUE 80.
           05  RECV-RETURN-LON-TYPE         PIC 9(6)    VALUE 81.
           05  SIDRC-RETURN-MLP-FORMAT      PIC 9(6)    VALUE 91.

      *-----------------------------------------------------------------*

      *    COPY TTETST3.
      *-----------------------------------------------------------------*
      * DISPLAY MESSAGES FOR TTPTST3
      *-----------------------------------------------------------------*
      * NORMAL MESSAGES
       01  MSG-NORMAL.
            05  MSG-START                  PIC  X(60) VALUE
             'ENTER START VALUES.'.
            05  MSG-END                    PIC  X(60) VALUE
             'TEST3  HAS ENDED.'.
            05  MSG-OK                     PIC  X(60) VALUE
             'FUNCTION COMPLETED - ENTER NEW REQUEST.'.
            05  MSG-RETURNING              PIC  X(60) VALUE
             'FUNCTION COMPLETED - ENTER NEW REQUEST.'.

            05  MSG-SYSTEM-INACTIVE        PIC  X(60) VALUE
             '       QUEUING SYSTEM IS NOT ACTIVE'.

      * ERROR  MESSAGES
       01  MSG-ERROR.
```

```
      05  MSG-ERR-QUEUE               PIC  X(60)  VALUE
       'QUEUE NAME NOT ENTERED.'.
      05  MSG-ERR-TS                  PIC  X(60)  VALUE
       'TIME STAMP FLAG MUST BE SPACE OR Y.'.
      05  MSG-ERR-MSG                 PIC  X(60)  VALUE
       'TEXT MESSAGE NOT ENTERED.'.
      05  MSG-ERR-MSG-SIZE            PIC  X(60)  VALUE
       'TEXT MESSAGE SIZE NOT ENTERED.'.
      05  MSG-ERR-MSG-SIZE-VALUE      PIC  X(60)  VALUE
       'TEXT MESSAGE SIZE IF INVALID.'.
      05  MSG-ERR-NUM-MSG             PIC  X(60)  VALUE
       'NUMBER OF MESSAGES TO BE PUT PER TASK NOT ENTERED.'.
      05  MSG-ERR-NUM-MSG-VALUE       PIC  X(60)  VALUE
       'NUMBER OF MESSAGES TO BE PUT PER TASK IS INVALID.'.
      05  MSG-ERR-MAX-TASK            PIC  X(60)  VALUE
       'NUMBER OF TASKS TO START NOT ENTERED.'.
      05  MSG-ERR-MAX-TASK-VALUE      PIC  X(60)  VALUE
       'NUMBER OF TASKS TO START IS INVALID.'.

      05  MSG-ERR-FUNCTION            PIC  X(60)  VALUE
       'FUNCTION NOT ENTERED.'.
      05  MSG-ERR-FUNCTION-VALUE      PIC  X(60)  VALUE
       'FUNCTION MUST BE A "G" OR "P".'.

      05  MSG-ERR-PFKEY               PIC  X(60)  VALUE
       'INVALID PFKEY WAS ENTERED  - ENTER VALID ONE.'.
      05  MSG-ERR-MAPFAIL             PIC  X(60)  VALUE
       'TASK ENTERED IMPROPERLY    - TASK RE-STARTED.'.

      05  MSG-ERR-MAPFAIL-REPEATED    PIC  X(60)  VALUE
       'TASK HAS REPEATED ERRORS - PLEASE CONTACT SUPPORT.'.

*--MAJOR ERROR THAT ARE LOGGED
      05  MSG-ERR-CICS                PIC  X(60)  VALUE
       'CICS ERROR          - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-TRANS-ID            PIC  X(60)  VALUE
       'OPTION NOT AVAILABLE- PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-NOFILE              PIC  X(60)  VALUE
       'CICS FILE  ERROR    - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-DISABLED            PIC  X(60)  VALUE
       'CICS DISABLE  ERROR - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-ILLOGIC             PIC  X(60)  VALUE
       'CICS ILLOGIC  ERROR - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-INVREQ              PIC  X(60)  VALUE
       'CICS REQUEST  ERROR - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-IOERR               PIC  X(60)  VALUE
       'CICS I/O      ERROR - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-NOTFOUND            PIC  X(60)  VALUE
       'CICS NOTFOUND ERROR - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-NOTOPEN             PIC  X(60)  VALUE
       'CICS NOTOPEN  ERROR - PLEASE CONTACT SUPPORT.'.
      05  MSG-ERR-ABENDED             PIC  X(60)  VALUE
       'CICS ABEND    ERROR - PLEASE CONTACT SUPPORT.'.

      05  MSG-ERR-USER-NOT-AUTH       PIC  X(60)  VALUE
       'USER IS NOT AUTHORIZED TO PERFORM FUNCTION. '.

*-------------------------------------------------------------*

*    COPY        MQWEOWS.
*-------------------------------------------------------------*
*Debugging eyecatcher information for end of WORKING-STORAGE.
*-------------------------------------------------------------*
 01  FILLER                      PIC X(16)  VALUE
      '================'.
 01  WS-RWS-PROGRAM-NAME2        PIC X(8).
 01  FILLER                      PIC X(16) VALUE
```

TTPTST3.Z

```
      ' Version V2.1.0'.
*-------------------------------------------------------------*
*-------------------------------------------------------------*
 LINKAGE SECTION.
*-------------------------------------------------------------*
 01  DFHCOMMAREA.
     05  FILLER                PIC X(400).


*--STARTED DATA
 01  LK-GET-DATA.
     05  FILLER                PIC X(400).


*-------------------------------------------------------------*
 PROCEDURE DIVISION.
*-------------------------------------------------------------*
 0000-MAIN.

*--SETUP ENVIRONMENT FROM LAST TIME
     PERFORM  1000-INITIAL.

*-- --IF  RECIEVEING - PROCESS FUNCTION
     IF MTP-SCREEN-RECEIVE
      THEN
            PERFORM  2000-SCREEN-FUNCTION
               THRU  2000-SCREEN-EXIT.
*
*-------------------------------------------------------------*
 0000-RETURN-MQMS.
     PERFORM 7000-SEND-MAP.
     MOVE 'R'          TO MTP-SCREEN-IND.
*
     EXEC  CICS RETURN TRANSID(MTP-ACTIVE-TASK)
                          COMMAREA(MTP-COMMAREA)
                 LENGTH  (LENGTH OF MTP-COMMAREA)
           END-EXEC.
*
     GOBACK.
  EJECT
*-------------------------------------------------------------*
 1000-INITIAL.
*-------------------------------------------------------------*
* PURPOSE: SETUP HANDLES
*          CHECK IF ENVIRONMENT EXIST - ALREADY
*          IF FIRST TIME - JUST SET MAIN SCREEN AND GET OUT
*-------------------------------------------------------------*
*
     EXEC CICS HANDLE CONDITION
                      ERROR   (9900-HANDLE-ERROR)
                      TRANSIDERR   (9900-HANDLE-TRANSID)
                      MAPFAIL      (9900-HANDLE-MAPFAIL)
                      FILENOTFOUND (9900-HANDLE-NOFILE)
                      DISABLED (9900-HANDLE-DISABLE)
                      ILLOGIC (9900-HANDLE-ILLOGIC)
                      INVREQ  (9900-HANDLE-INVREQ)
                      IOERR   (9900-HANDLE-IOERR)
                      NOTFND  (9900-HANDLE-NOTFOUND)
                      NOTOPEN (9900-HANDLE-NOTOPEN)
          END-EXEC.

*--SET ERROR INFO
     PERFORM 1050-SET-ERROR-INFO.

*--GET WHAT SYSTEM / APPLIC IS RUNNING
     EXEC CICS ASSIGN SYSID (WS-SYSID)
                     APPLID (WS-APPLID)
```

```
                STARTCODE (WS-STARTCD)
          END-EXEC.

*--CHECK IF SYSTEM EXIST - ALREADY
     PERFORM 1100-CHECK-SYSTEM
        THRU 1100-EXIT.

*--SETUP  ENVIRONMENT
     PERFORM 1200-SETUP-ENVIR
        THRU 1200-EXIT.

*-------------------------------------------------------------*
 1000-EXIT.
     EXIT.
  EJECT
*-------------------------------------------------------------------*
 1050-SET-ERROR-INFO.
*-------------------------------------------------------------------*
* PURPOSE: SET DEFAULT ERROR INFO
*-------------------------------------------------------------------*
*--SET CSMT DATE AND TIME
     EXEC CICS ASKTIME
               ABSTIME(WS-ABSTIME)
          END-EXEC.

     MOVE EIBTIME           TO WS-UNPACK-TIME-9.
     MOVE WS-TIME-HH        TO WS-FORMAT-TIME-HH
     MOVE WS-TIME-MM        TO WS-FORMAT-TIME-MM.
     MOVE WS-TIME-SS        TO WS-FORMAT-TIME-SS.

     EXEC CICS FORMATTIME
               ABSTIME (WS-ABSTIME)
               MMDDYY  (WS-FORMATTED-DATE)
               DATESEP ('/')
          END-EXEC.
*-- Save the year field which is held in the century field
*   immediately after the FORMATTIME
     MOVE WS-FORMAT-DATE-CC TO WS-FORMAT-DATE-YY
*
     EXEC  CICS FORMATTIME
                ABSTIME(WS-ABSTIME)
                YYMMDD (WS-DATE-YYMMDD)
           END-EXEC.

*-- --SET CENTURY
     IF WS-DATE-YY >  60
      THEN
             MOVE  19                 TO  WS-DATE-CC
                                          WS-FORMAT-DATE-CC
      ELSE
             MOVE  20                 TO  WS-DATE-CC
                                          WS-FORMAT-DATE-CC
      END-IF

*--SET COMMON ERROR INFO
     MOVE  ZERO         TO  ERR-CODE.
     MOVE  'TTPTST3'    TO  ERR-PROGRAM.

*-------------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 1100-CHECK-SYSTEM.
*-------------------------------------------------------------*
* PURPOSE: LINK TO FINQ TO GET SYSTEM STATUS
*-------------------------------------------------------------*
*--SET UP COMMAREA
     MOVE SPACES   TO FINDQ-CALL-PARAMETERS.
```

```
     MOVE 'S'      TO FINDQ-CALL-TYPE.

*--CALL
     EXEC CICS LINK PROGRAM (ENV-II-LINK-FINDQ)
                    COMMAREA (FINDQ-CALL-PARAMETERS)
           LENGTH(LENGTH OF  FINDQ-CALL-PARAMETERS)
          END-EXEC.
*
*-------------------------------------------------------------*
 1100-EXIT.
     EXIT.
  EJECT
*-------------------------------------------------------------*
 1200-SETUP-ENVIR.
*-------------------------------------------------------------*
* PURPOSE: SETUP PROGRAM ENVIR
*-------------------------------------------------------------*
*--SETUP NEW COMMON AREA
     MOVE LOW-VALUES  TO MAINO.

*--IF NOT RE-STARTED
     IF  NOT WS-STARTED
      THEN
*-- --IF NOT STARTED AND NO COMMAREA - JUST SETUP TO MAIN...
          IF  (EIBCALEN EQUAL  ZERO)
           THEN
                 MOVE 'S'         TO MTP-SCREEN-IND
                 MOVE MSG-START   TO WS-ERROR-MESSAGE
           ELSE
*-- --MOVE COMMAREA TO WORKING-STORAGE..CONTINUE
                 MOVE DFHCOMMAREA  TO MTP-COMMAREA
                 MOVE 'R'          TO MTP-SCREEN-IND
          END-IF.


*--STARTED - TREAT AS NEW TASK
     IF WS-STARTED
      THEN
          PERFORM 1210-GET-STARTED-DATA
             THRU 1210-GET-STARTED-EXIT

*-- --IF RETURNING FROM ANOTHER APPLI. - TREAT AS NEW
          MOVE LOW-VALUES     TO MAINO
          IF  MTP-SCREEN-RETURN
           THEN
               MOVE MSG-RETURNING  TO WS-ERROR-MESSAGE
               MOVE MTP-CONFIG-DATA
                                   TO MTP-MAIN-TASK
               MOVE SPACES         TO MTP-CONFIG-DATA
           ELSE
               MOVE MSG-START      TO WS-ERROR-MESSAGE
          END-IF
          MOVE 'S'            TO MTP-SCREEN-IND.

*--SETUP TASK ID
     MOVE EIBTRNID  TO   MTP-ACTIVE-TASK.

*-------------------------------------------------------------*
 1200-EXIT.
     EXIT.
  EJECT
*-------------------------------------------------------------*
 1210-GET-STARTED-DATA.
*-------------------------------------------------------------*
* PURPOSE: READ STARTED DATA
*-------------------------------------------------------------*
*--GET
```

```
     EXEC CICS RETRIEVE
               SET    (ADDRESS OF LK-GET-DATA)
               LENGTH (WS-REC-SIZE)
          END-EXEC.
*
     IF WS-REC-SIZE  NOT <  LENGTH OF MTP-COMMAREA
      THEN
*-- --GOT VALID LENGTH- MOVE AND CHECK
            MOVE LK-GET-DATA  TO MTP-COMMAREA
            IF NOT   MTP-HEADER-OK
*-- -- --ERROR IN GET DATA - RESET COMMAREA
             THEN
                 MOVE SPACES  TO MTP-COMMAREA
                 SET             MTP-HEADER-OK  TO TRUE
                 MOVE 'S'     TO MTP-SCREEN-IND
                 MOVE 'MAIN'  TO MTP-MAP-VALUE.
*
*-------------------------------------------------------------*
 1210-GET-STARTED-EXIT.
     EXIT.
  EJECT
*-------------------------------------------------------------*
 2000-SCREEN-FUNCTION.
*-------------------------------------------------------------*
* PURPOSE: GET MAIN MAP
*          CHECK OPTION KEYS
*          CHECK OPTION FIELD
*          PROCESS FUNCTION ENTERED
*-------------------------------------------------------------*
*--PRELIMINARY EDIT OF PF KEYS
     PERFORM  2100-MAIN-CHECK-KEYS.
     IF  NOT WS-EDIT-ERR
      THEN

*--GET MAP
            PERFORM  7000-RECEIVE-MAP

*--IF RECORD NOT FOUND - SET UP DEFAULT RECORD
            IF NOT FINDQ-SYSTEM-ACTIVE
             THEN
                  MOVE MSG-SYSTEM-INACTIVE
                            TO WS-ERROR-MESSAGE
             ELSE
*-- --EDIT MAP
                  PERFORM  2200-MAIN-EDIT
                     THRU  2200-MAIN-EXIT

*--PROCESS FUNCTION KEY - IF NO ERRORS
                  IF  NOT WS-EDIT-ERR
                   THEN
                       PERFORM  2300-MAIN-FUNCTION
                          THRU  2300-MAIN-EXIT.

*-------------------------------------------------------------*
 2000-SCREEN-EXIT.
     EXIT.
  EJECT
*-------------------------------------------------------------*
 2100-MAIN-CHECK-KEYS.
*-------------------------------------------------------------*
* PURPOSE: PRELIMINARY PF KEY CHECK
*-------------------------------------------------------------*
*--CHECK AID KEY
*-- --MAIN MENU
     IF  (EIBAID     EQUAL  DFHPF2)
     AND (MTP-MAIN-TASK  NOT EQUAL  SPACES)
      THEN
```

```
           GO TO  9000-MAIN-MENU.

*-- --SHUTDOWN
     IF  ((EIBAID  EQUAL  DFHCLEAR OR DFHPA1 OR DFHPA2)
     OR   (EIBAID  EQUAL  DFHPF3))
      THEN
           GO TO  9000-SHUTDOWN.

*-- --QUEUE KEYS - FIRST INQ THEN UPDATE
     IF (EIBAID     EQUAL  DFHPF4)
     OR (EIBAID     EQUAL  DFHENTER)
      THEN
          NEXT SENTENCE
      ELSE
          MOVE -1       TO LTNUML
          MOVE 'Y'     TO WS-EDIT-ERR-FLAG
          MOVE MSG-ERR-PFKEY    TO WS-ERROR-MESSAGE.


*--SET TYPE OF FUNCTION - DEFAULT TO UPDATE
     MOVE  'UPDATE'   TO MTP-MAP-FUNCTION.
*
*-----------------------------------------------------------------*
  EJECT
*-----------------------------------------------------------------*
 2200-MAIN-EDIT.
*-----------------------------------------------------------------*
* PURPOSE: EDIT SCREEN
*-----------------------------------------------------------------*

*--FUNCTION
     MOVE DFHBMFSE   TO LFUNCA.
     IF (LFUNCI EQUAL    '?')
     OR (LFUNCI NOT >    SPACE)
      THEN
           MOVE '?'      TO LFUNCO
           MOVE -1       TO LFUNCL
           MOVE DFHUNIMD TO LFUNCA
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-FUNCTION
                          TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
      ELSE
           IF (LFUNCI   EQUAL  'P')
            THEN
                 MOVE  'PUT'   TO TST2-FUNCTION
            ELSE
           IF (LFUNCI   EQUAL  'G')
            THEN
                 MOVE  'GET'   TO TST2-FUNCTION
            ELSE
                    MOVE -1       TO LFUNCL
                    MOVE DFHUNIMD TO LFUNCA
                    MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                    MOVE MSG-ERR-FUNCTION-VALUE
                                  TO WS-ERROR-MESSAGE
                    PERFORM  8000-MOVE-ERR-MESSAGE.


*--NUMBER OF STARTS
     MOVE DFHBMFSE   TO LTNUMA.
     IF (LTNUMI EQUAL    '?')
     OR (LTNUMI NOT >    SPACE)
      THEN
           MOVE '?'      TO LTNUMO
           MOVE -1       TO LTNUML
           MOVE DFHUNIMD TO LTNUMA
```

```
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-MAX-TASK
                         TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
      ELSE
           IF (LTNUMI     NUMERIC)
            THEN
                 MOVE  LTNUMI TO WS-NUM
                 IF (WS-NUM    <   0)
                  THEN
                    MOVE -1       TO LTNUML
                    MOVE DFHUNIMD TO LTNUMA
                    MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                    MOVE MSG-ERR-MAX-TASK-VALUE
                                  TO WS-ERROR-MESSAGE
                    PERFORM  8000-MOVE-ERR-MESSAGE
                  ELSE
                    MOVE WS-NUM TO WS-SS-STARTS
            ELSE
                 MOVE -1       TO LTNUML
                 MOVE DFHUNIMD TO LTNUMA
                 MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                 MOVE MSG-ERR-MAX-TASK-VALUE
                               TO WS-ERROR-MESSAGE
                 PERFORM  8000-MOVE-ERR-MESSAGE.

*--CHECK QUEUE FIELD
     MOVE DFHBMFSE   TO LPQUEA.
     IF (LPQUEI EQUAL    '?')
     OR (LPQUEI NOT >    SPACE)
      THEN
           MOVE '?'      TO LPQUEO
           MOVE -1       TO LPQUEL
           MOVE DFHUNIMD TO LPQUEA
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-QUEUE
                         TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
      ELSE
           MOVE  LPQUEI TO TST2-PUT-QUEUE-NAME.

*--NUM OF MESSAGE PER TASK
     MOVE DFHBMFSE   TO LMNUMA.
     IF (LMNUMI EQUAL    '?')
     OR (LMNUMI NOT >    SPACE)
      THEN
           MOVE '?'      TO LMNUMO
           MOVE -1       TO LMNUML
           MOVE DFHUNIMD TO LMNUMA
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-NUM-MSG
                         TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
      ELSE
           IF (LMNUMI     NUMERIC)
            THEN
                 MOVE  LMNUMI TO WS-NUM
                 IF (WS-NUM    <   0)
                  THEN
                    MOVE -1       TO LMNUML
                    MOVE DFHUNIMD TO LMNUMA
                    MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                    MOVE MSG-ERR-NUM-MSG-VALUE
                                  TO WS-ERROR-MESSAGE
                    PERFORM  8000-MOVE-ERR-MESSAGE
                  ELSE
                    MOVE WS-NUM TO TST2-PUT-NUM-MSG
```

```
            ELSE
                  MOVE -1       TO LMNUML
                  MOVE DFHUNIMD TO LMNUMA
                  MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                  MOVE MSG-ERR-NUM-MSG-VALUE
                                TO WS-ERROR-MESSAGE
                  PERFORM  8000-MOVE-ERR-MESSAGE.


*--MESSAGE SIZE
     MOVE DFHBMFSE   TO LPSIZEA.
     IF TST2-FUNCT-PUT
      THEN
        IF  ((LPSIZEI EQUAL    '?')  OR (LPSIZEI NOT >    SPACE))
         THEN
           MOVE '?'      TO LPSIZEO
           MOVE -1       TO LPSIZEL
           MOVE DFHUNIMD TO LPSIZEA
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-MSG-SIZE
                          TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
         ELSE
           IF (LPSIZEI     NUMERIC)
            THEN
                  MOVE  LPSIZEI TO WS-NUM
                  IF (WS-NUM    <   0)  OR
                     (WS-NUM    >  32000)
                   THEN
                     MOVE -1       TO LPSIZEL
                     MOVE DFHUNIMD TO LPSIZEA
                     MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                     MOVE MSG-ERR-MSG-SIZE-VALUE
                                   TO WS-ERROR-MESSAGE
                     PERFORM  8000-MOVE-ERR-MESSAGE
                   ELSE
                     MOVE WS-NUM TO TST2-PUT-MSG-SIZE
            ELSE
                  MOVE -1       TO LPSIZEL
                  MOVE DFHUNIMD TO LPSIZEA
                  MOVE 'Y'      TO WS-EDIT-ERR-FLAG
                  MOVE MSG-ERR-MSG-SIZE-VALUE
                                TO WS-ERROR-MESSAGE
                  PERFORM  8000-MOVE-ERR-MESSAGE.

*--CHECK MESSAGE
     MOVE DFHBMFSE   TO LMSGA.
     IF TST2-FUNCT-PUT
      THEN
       IF (LMSGI EQUAL    '?')  OR (LMSGI NOT >    SPACE)
        THEN
           MOVE '?'      TO LMSGO
           MOVE -1       TO LMSGL
           MOVE DFHUNIMD TO LMSGA
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-MSG
                          TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
        ELSE
           MOVE  LMSGI TO TST2-PUT-MSG.

*--CHECK TIME STAMP FLAG
     MOVE DFHBMFSE   TO LTSA.
     IF TST2-FUNCT-PUT
      THEN
        IF (LTSI EQUAL    '?')  OR (LTSI NOT >    SPACE)
         THEN
```

```
           MOVE '?'      TO LTSO
           MOVE -1       TO LTSL
           MOVE DFHUNIMD TO LTSA
           MOVE 'Y'      TO WS-EDIT-ERR-FLAG
           MOVE MSG-ERR-TS
                          TO WS-ERROR-MESSAGE
           PERFORM  8000-MOVE-ERR-MESSAGE
         ELSE
           IF TST2-PUT-MSG-TIMESTAMP EQUAL SPACE OR 'Y'
            THEN
                 MOVE  LTSI TO TST2-PUT-MSG-TIMESTAMP.


*---------------------------------------------------------------*
 2200-MAIN-EXIT.
     EXIT.
  EJECT
*---------------------------------------------------------------*
 2300-MAIN-FUNCTION.
*---------------------------------------------------------------*
* PURPOSE: SETUP DEFAULT RECORD AND MESSAGE
*          DEFAULT TO QUEUE PROCESSING
*---------------------------------------------------------------*
*--SET CURSOR
     MOVE -1             TO LTNUML.


*--START TASK.
     PERFORM  WS-SS-STARTS  TIMES
          EXEC CICS START TRANSID('TST2')
                   INTERVAL (000000)
                      FROM  (WS-TST2-COMMAREA)
                    LENGTH  (LENGTH OF WS-TST2-COMMAREA)
             END-EXEC
     END-PERFORM.

*--SAYS OK
     MOVE MSG-OK  TO WS-ERROR-MESSAGE.

*---------------------------------------------------------------*
 2300-MAIN-EXIT.
     EXIT.
  EJECT
     EJECT
*---------------------------------------------------------------*
 7000-RECEIVE-MAP.
*---------------------------------------------------------------*
* PURPOSE: GET USER MAP
*---------------------------------------------------------------*
     EXEC CICS RECEIVE MAP (MTP-MAP-VALUE)
                    MAPSET('TTMTST3')
                     INTO (MAINO)
          END-EXEC.
*
*---------------------------------------------------------------*
     EJECT
*---------------------------------------------------------------*
 7000-SEND-MAP.
*---------------------------------------------------------------*
* PURPOSE: SETUP HEADER DATA
*          SEND SCREEN BASED ON MODE
*---------------------------------------------------------------*
*--SETUP HEADER
     PERFORM 7100-SETUP-HEADER.

*--RESET ERROR TO FIRST ONE..IF MORE THAN ONE
     IF WS-ERR-COUNT  > ZERO
```

```
      THEN
             MOVE  WS-ERR-MSG (1)
                       TO WS-ERROR-MESSAGE.
*
*--SEND SCREEN
     IF MTP-SCREEN-SEND
      THEN
*-- --NEW MAP - SETUP INFO....
         MOVE  WS-ERROR-MESSAGE  TO  LERRO
         EXEC CICS SEND MAP (MTP-MAP-VALUE)
                     MAPSET('TTMTST3')
                      FROM (MAINO)
                       ERASE CURSOR
              END-EXEC
      ELSE
         MOVE  WS-ERROR-MESSAGE  TO  LERRO
         EXEC CICS SEND MAP (MTP-MAP-VALUE)
                     MAPSET('TTMTST3')
                      FROM (MAINO)
                       DATAONLY CURSOR
              END-EXEC.

*-------------------------------------------------------------*
     EJECT
*-------------------------------------------------------------*
 7100-SETUP-HEADER.
*-------------------------------------------------------------*
* PURPOSE: SETUP HEADER DATA
*-------------------------------------------------------------*
*--SETUP HEADER
     MOVE WS-FORMATTED-DATE  TO MDATELO.
     MOVE DFHBMPRF           TO MDATELA.
     MOVE WS-FORMATTED-TIME  TO MTIMELO.

     MOVE WS-SYSID   TO MSYSTLO.
     MOVE EIBTRMID   TO MTERMLO.
     MOVE WS-APPLID  TO MAPPLLO.


*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 8000-MOVE-ERR-MESSAGE.
*-------------------------------------------------------------*
* PURPOSE: MOVE MULTIPLE ERROR MESSAGES...
*-------------------------------------------------------------*
     ADD +1  TO WS-ERR-COUNT.
     IF WS-ERR-COUNT NOT > WS-ERR-MAX
      THEN
             MOVE WS-ERROR-MESSAGE
                       TO  WS-ERR-MSG (WS-ERR-COUNT).

*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 9000-SHUTDOWN.
*-------------------------------------------------------------*
* PURPOSE: SHUTDOWN PROGRAM
*-------------------------------------------------------------*
*--IF ORIGIN TRAN WAS ME ............
     EXEC CICS SEND FROM (MSG-END)
        LENGTH (LENGTH OF MSG-END) ERASE
                 END-EXEC.

*
     EXEC CICS RETURN
                 END-EXEC.
```

```
*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 9000-MAIN-MENU.
*-------------------------------------------------------------*
* PURPOSE: RETURN TO MAIN TASK
*-------------------------------------------------------------*
*--RE-START ORIGINAL TASK
     MOVE SPACE  TO  MTP-SCREEN-IND.
     EXEC  CICS START TRANSID(MTP-MAIN-TASK)
                      TERMID(EIBTRMID)
                      FROM  (MTP-COMMAREA)
            LENGTH(LENGTH OF  MTP-COMMAREA)
                     INTERVAL(0)
                     NOHANDLE
           END-EXEC.

*
     EXEC CICS RETURN
                 END-EXEC.

*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
* PURPOSE: ENVIRONMENT NOT SETUP
*-------------------------------------------------------------*
 9900-NO-ENVIR-SETUP.
     EXEC CICS SEND FROM (ENV-IT-UN-INIT-MSG)
        LENGTH (LENGTH OF ENV-IT-UN-INIT-MSG) ERASE
       END-EXEC

     EXEC CICS RETURN
      END-EXEC.

  EJECT
*-------------------------------------------------------------*
* PURPOSE: ERROR CONDITION
*-------------------------------------------------------------*
 9900-HANDLE-TRANSID.
     MOVE ERR-CICS-TRANIDERR     TO ERR-CODE.
     MOVE WS-TRAN-ID             TO ERR-DETAIL.
     MOVE MSG-ERR-TRANS-ID       TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-NOTAUTH.
     MOVE ERR-PROC-NOT-AUTHORIZED   TO ERR-CODE.
     MOVE WS-TRAN-ID                TO ERR-DETAIL.
     MOVE MSG-ERR-USER-NOT-AUTH     TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-ERROR.
     MOVE ERR-CICS-ERROR    TO ERR-CODE.
     MOVE MSG-ERR-CICS      TO WS-ERROR-MESSAGE.
     GO TO  9999-FATAL-ERR-EXIT.


 9900-HANDLE-NOFILE.
     MOVE ERR-CICS-NOFILE  TO ERR-CODE.
     MOVE MSG-ERR-NOFILE   TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-DISABLE.
     MOVE  ERR-CICS-DISABLE     TO ERR-CODE.
     MOVE  MSG-ERR-DISABLED        TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.
```

```
 9900-HANDLE-ILLOGIC.
     MOVE ERR-CICS-ILLOGIC  TO ERR-CODE.
     MOVE MSG-ERR-ILLOGIC     TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-INVREQ.
     MOVE ERR-CICS-INVALID-REQ  TO ERR-CODE.
     MOVE MSG-ERR-INVREQ        TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-IOERR.
     MOVE ERR-CICS-IO-ERROR   TO ERR-CODE.
     MOVE MSG-ERR-IOERR         TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-NOTFOUND.
     MOVE ERR-LOGIC-MISSING-RECORD TO ERR-CODE.
     MOVE MSG-ERR-NOTFOUND         TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.

 9900-HANDLE-NOTOPEN.
     MOVE  ERR-CICS-FILE-NOTOPEN   TO ERR-CODE.
     MOVE  MSG-ERR-NOTOPEN           TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.


 9900-HANDLE-MAPFAIL.
     EXEC CICS HANDLE CONDITION
                     MAPFAIL (9999-FATAL-ERR-PRE-EXIT)
            END-EXEC.

     MOVE  ERR-CICS-MAPFAIL     TO ERR-CODE.
     MOVE  MSG-ERR-MAPFAIL      TO WS-ERROR-MESSAGE.
     GO TO  9900-ERR-EXIT.
  EJECT
*-------------------------------------------------------------*
 9900-ERR-EXIT.
*-------------------------------------------------------------*
* PURPOSE: ERROR CONDITION
*          SEND SCREEN
*          GO TO CICS RETURN W/ NEXT TRAN ID
*-------------------------------------------------------------*
*--TRANSLATE ERROR CODE
     PERFORM 9999-CONVERT-ERROR-INFO.

*--WRITE ERROR MESSAGE
     PERFORM 9999-ERROR-WRITE.

*--RE-SEND MAIN MAP
     MOVE LOW-VALUES  TO MAINO.
     MOVE -1          TO LTNUML.
     MOVE 'F'         TO MTP-SCREEN-IND.

     GO TO  0000-RETURN-MQMS.
  EJECT
*-------------------------------------------------------------*
 9999-FATAL-ERR-PRE-EXIT.
*-------------------------------------------------------------*
* PURPOSE: REPEATED MAPFAIL
*-------------------------------------------------------------*
*--SET ERROR MESSAGE
     MOVE   MSG-ERR-MAPFAIL-REPEATED  TO  WS-ERROR-MESSAGE.
     GO TO  9999-FATAL-ERR-EXIT.


*-------------------------------------------------------------*
 9999-FATAL-ERR-EXIT.
```

```
      *-------------------------------------------------------------*
      * PURPOSE: ERROR EXIT - FOR REPEATED MAPFAIL / ABEND
      *-------------------------------------------------------------*
      *--SEND MESSAGE
           EXEC CICS SEND FROM (WS-ERROR-MESSAGE)
              LENGTH (LENGTH OF WS-ERROR-MESSAGE) ERASE NOHANDLE
             END-EXEC.

      *--GET OUT
           EXEC CICS RETURN
               END-EXEC.
        EJECT
      *-------------------------------------------------------------*
      * ERROR HANDLING CODE
      *-------------------------------------------------------------*
      *     COPY        MQIERRCD.
/INCLUDE MQIERRCD

      *--ABEND MESSAGE SENT....JUST GET OUT
           MOVE   MSG-ERR-ABENDED  TO  WS-ERROR-MESSAGE.
           GO TO  9999-FATAL-ERR-EXIT.
```

Sample program MQPECHO.Z

```
       COPY COPYRSAP.
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    MQPECHO.
       AUTHOR.        IBM.
      ****************************************************************
      *                T E S T     ECHO                              *
      *                                                              *
      *           A P P L I C A T I O N   I N T E R F A C E          *
      *                                                              *
      *                MQSeries for VSE/ESA                          *
      *                                                              *
      *--------------------------------------------------------------*
      * MQPECHO - IBM APPLICATION TEST PROGRAM TO ECHO MESSAGE FROM   *
      *           A QUEUE, SAY XXX, TO A QUEUE NAMED 'IBM.REPLY.QUEUE'.*
      *                                                              *
      * PREREQUISITE:                                                *
      *           1. SENDING QUEUE, A LOCAL QUEUE NAMED XXX, MUST BE  *
      *              DEFINED WITH                                    *
      *                   TRIGGER ENAABLE: Y                         *
      *                   PROGRAM ID     : MQPECHO                   *
      *           2. SENDING QUEUE MUST BE ABLE TO TRIGGER MQPECHO    *
      *              A. IF XXX HAS MESSAGES, STOP THEN START XXX      *
      *              B. IF XXX DOESN'T HAVE ANY MESSAGES OR YOU WANT TO*
      *                 ECHO MORE MESSAGES THAN EXISTING ONES, THEN PUT*
      *                 SOME MESSAGES BY, EG, TST1 PUT  99 XXX       *
      *           3. DEFINE IBM.REPLY.QUEUE, IF IT DOES NOT EXIST    *
      *                                                              *
      * FUNCTIONS:  1. ACTIVATED VIA TRIGGER MECHANISM BY QUEUE XXX.  *
      *             2. READ QUEUE XXX TILL THERE IS NO MORE MESSAGE   *
      *             3. ECHO READ MESSAGE INTO IBM.REPLY.QUEUE        *
      *                                                              *
      * COPYBOOKS:  MQIVALUE - IBM RETURN CODES.                     *
      *             MQIERR   - ERROR COMMAREA                        *
      *             MQIERRC  - ERROR COMMON CODES                    *
      *             MQIERRCD - ERROR CODE                            *
      *             MQICENV  - ENVIRONMENT                           *
      *                                                              *
      * CALLS    :  MQCONN   - CONNECT                               *
      *             MQOPEN   - OPEN                                  *
      *             MQPUT    - PUT                                   *
      *             MQGET    - GET                                   *
      *             MQCLOSE  - CLOSE                                 *
      *             MQDISC   - DISCONNECT                            *
```

```
*                                                               *
* CALLED BY:  -- NONE --                                        *
*                                                               *
* CHANGE SUMMARY:                                               *
*                                                               *
*---------------------------------------------------------------*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 DATA DIVISION.
*---------------------------------------------------------------*

 WORKING-STORAGE SECTION.
*     COPY COPYRWS.
*---------------------------------------------------------------*
* COPYRIGHT WORKING STORAGE  FOR COBOL MODULES                   *
*---------------------------------------------------------------*
 01   FILLER.
      05 FILLER                 PIC X(72) VALUE
      'Licensed Materials - Property of IBM'.
      05 FILLER                 PIC X(72) VALUE SPACES.
      05 FILLER                 PIC X(72) VALUE
      '5686-A06  '.
      05 FILLER                 PIC X(72) VALUE SPACES.
      05 FILLER                 PIC X(72) VALUE
      '(C) Copyright IBM Corp. 1998, 2002       All Rights Reserve
-     'd'.
      05 FILLER                 PIC X(72) VALUE SPACES.
      05 FILLER                 PIC X(72) VALUE
      'US Government Users Restricted Rights - Use, duplication or
-     '  '.
      05 FILLER                 PIC X(72) VALUE
      'disclosure restricted by GSA ADP Schedule Contract with IBM
-     ' Corp.'.
*---------------------------------------------------------------*
*Debugging eyecatcher information for start of WORKING-STORAGE.
*---------------------------------------------------------------*
 01  WS-RWS-PROGRAM-NAME1        PIC X(8).
 01  FILLER                      PIC X(16) VALUE
      ' Version V2.1.0'.
 01  WS-RWS-WHEN-COMPILED        PIC X(21).
 01  FILLER                      PIC X(7)  VALUE '======='.
*---------------------------------------------------------------*
*---------------------------------------------------------------*

      COPY MQWTRACE.


 01  WS-WORK-FIELDS.
     05  WS-MORE-FLAG              PIC  XX    VALUE SPACES.
       88 WS-MORE-DATA                        VALUE SPACES.
       88 WS-NOMORE-DATA                      VALUE 'Y'.

     05  WS-DATA-LENGTH            PIC S9(4)  COMP   VALUE ZERO.
     05  WS-APPL-MSG-LENGTH        PIC S9(8)  COMP   VALUE ZERO.
     05  WS-ABSTIME               PIC S9(15) COMP-3.
     05  WS-DATE.
        10  WS-DATE-CC            PIC  99  VALUE ZERO.
        10  WS-DATE-YYMMDD.
          12  WS-DATE-YY          PIC  99  VALUE ZERO.
          12  WS-DATE-MM          PIC  99  VALUE ZERO.
          12  WS-DATE-DD          PIC  99  VALUE ZERO.
          12  FILLER              PIC  XX  VALUE ZERO.

     05  WS-UNPACK-TIME-9         PIC 9(07) VALUE ZEROES.
     05  WS-UNPACK-TIME-X REDEFINES WS-UNPACK-TIME-9.
         10  FILLER              PIC  X(01).
         10  WS-TIME-HHMMSS.
```

```
            12  WS-TIME-HH         PIC  X(02).
            12  WS-TIME-MM         PIC  X(02).
            12  WS-TIME-SS         PIC  X(02).
     05  WS-FORMATTED-TIME.
         10  WS-FORMAT-TIME-HH     PIC  X(02) VALUE SPACES.
         10  FILLER                PIC  X(01) VALUE ':'.
         10  WS-FORMAT-TIME-MM     PIC  X(02) VALUE SPACES.
         10  FILLER                PIC  X(01) VALUE ':'.
         10  WS-FORMAT-TIME-SS     PIC  X(02) VALUE SPACES.
     05  WS-FORMATTED-DATE.
         10  WS-FORMAT-DATE-MM     PIC  X(02) VALUE SPACES.
         10  FILLER                PIC  X(01) VALUE '/'.
         10  WS-FORMAT-DATE-DD     PIC  X(02) VALUE SPACES.
         10  FILLER                PIC  X(01) VALUE '/'.
         10  WS-FORMAT-DATE-YY     PIC  X(02) VALUE SPACES.

*--DEFAULT ECHO READQUEUE/QM
     05  WS-READ-QM-QUEUE.
       10  WS-QM-NAME               PIC  X(48) VALUE SPACES.
       10  WS-Q-NAME                PIC  X(48) VALUE
       'QUEUE1'.

*--DEFAULT ECHO RESPONSE QUEUE/QM
     05  WS-RESPONSE-QM-QUEUE.
       10  WS-R-QM-NAME               PIC  X(48) VALUE SPACES.
       10  WS-R-Q-NAME                PIC  X(48) VALUE
       'IBM.REPLY.QUEUE'.

*-------------------------------------------------------------*

*-------------------------------------------------------------*
* ERROR MESSAGE FOR QUEUE
*-------------------------------------------------------------*
 01  WS-ERROR-MESSAGE.
     05 FILLER                     PIC  X(5) VALUE
     'ECHO:'.
     05 FILLER                     PIC  X(6)  VALUE
     ' QID -'.
     05 WS-ERR-DISPLAY-QUEUE       PIC  X(30) VALUE  SPACES.
     05 FILLER                     PIC  X(6)  VALUE
     ',CC  -'.
     05 WS-ERR-DISPLAY-CCODE       PIC  9(4)  VALUE  ZERO.
     05 FILLER                     PIC  X(6)  VALUE
     ',RC  -'.
     05 WS-ERR-DISPLAY-RCODE       PIC  9(4)  VALUE  ZERO.

     05 WS-FUNCTION                PIC  X(12) VALUE  SPACES.

*
*-------------------------------------------------------------*

*-------------------------------------------------------------*
* ERROR WS VALUES
 01  WS-ERR-INFO.
*     COPY   MQIERR.
*-------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQIERR    ***          - BEGIN - *
*-------------------------------------------------------------*
*      ERROR MODULE CALLING PARAMETERS                         *
*-------------------------------------------------------------*

     02 ERR-HANDLER-COMMAREA.
        05 ERR-CURRENT-INFO.
          10 ERR-COM-HANDLER       PIC  X(48) VALUE SPACES.
          10 ERR-QUEUE             PIC  X(48) VALUE SPACES.
          10 ERR-FILE              PIC  X(8)  VALUE SPACES.
          10 ERR-DETAIL            PIC  X(80) VALUE SPACES.
```

```
           10 ERR-DETAIL2          PIC  X(80) VALUE SPACES.
           10 ERR-Q-CODE           PIC S9(8) COMP VALUE ZERO.
           10 FILLER               PIC  X(8) VALUE SPACES.

         05 ERR-RESULTS.
           10 ERR-CODE             PIC  9(6) VALUE ZERO.
           10 FILLER               PIC  XX   VALUE SPACES.
           10 ERR-PROGRAM          PIC  X(8) VALUE SPACES.
           10 ERR-TRANID           PIC  X(4) VALUE SPACES.
           10 ERR-TERMID           PIC  X(4) VALUE SPACES.
           10 ERR-TASKNO           PIC S9(7)  COMP-3 VALUE ZERO.
           10 ERR-ABSTIME          PIC S9(15) COMP-3 VALUE ZERO.

           10 ERR-DEBUG-EIBFN      PIC  XX   VALUE SPACES.
           10 ERR-DEBUG-EIBRCODE   PIC  X(6) VALUE LOW-VALUES.
           10 ERR-DEBUG-EIBRSRCE   PIC  X(8) VALUE LOW-VALUES.
           10 ERR-DEBUG-EIBRESP    PIC S9(8) COMP VALUE ZEROS.
           10 ERR-DEBUG-EIBRESP2   PIC S9(8) COMP VALUE ZEROS.
           10 ERR-DEBUG-EIBERRCD   PIC  X(4) VALUE LOW-VALUES.
           10 ERR-DEBUG-ABEND      PIC  X(4) VALUE SPACES.
           10 FILLER               PIC  X(12) VALUE SPACES.

*-----------------------------------------------------------------*
* - END -        ***  COPYBOOK: MQIERR    ***           - END - *
*-----------------------------------------------------------------*
*     COPY   MQIERRC.
*-----------------------------------------------------------------*
* IBM MQSERIES COMMON ERROR CODES
*-----------------------------------------------------------------*
 01  MSG-ERROR-MESSAGES.
     05 ERR-NO-ENVIRONMENT          PIC  9(6) VALUE 900000.

     05 ERR-CICS-ERROR              PIC  9(6) VALUE 800000.
     05 ERR-CICS-INVALID-REQ        PIC  9(6) VALUE 800010.
     05 ERR-CICS-ILLOGIC            PIC  9(6) VALUE 800011.
     05 ERR-CICS-ERROR-CHECKPOINT   PIC  9(6) VALUE 800090.
     05 ERR-CICS-ABEND              PIC  9(6) VALUE 800099.
     05 ERR-CICS-FILE-NOTOPEN       PIC  9(6) VALUE 801012.
     05 ERR-CICS-DISABLE            PIC  9(6) VALUE 801019.
     05 ERR-CICS-NO-STORAGE         PIC  9(6) VALUE 802000.
     05 ERR-CICS-LENGTH-ERR         PIC  9(6) VALUE 803001.
     05 ERR-CICS-MAPFAIL            PIC  9(6) VALUE 808000.
     05 ERR-CICS-PGMIDERR           PIC  9(6) VALUE 809000.
     05 ERR-CICS-FILEID             PIC  9(6) VALUE 809010.
     05 ERR-CICS-NOFILE             PIC  9(6) VALUE 809011.
     05 ERR-CICS-IO-ERROR           PIC  9(6) VALUE 809012.
     05 ERR-CICS-TRANIDERR          PIC  9(6) VALUE 809050.

     05 ERR-COM-FREE-ERROR          PIC  9(6) VALUE 501001.
     05 ERR-COM-EIB-ERROR           PIC  9(6) VALUE 501002.
     05 ERR-COM-STAT-ERROR          PIC  9(6) VALUE 501003.
     05 ERR-COM-ALLOC-ERROR         PIC  9(6) VALUE 501004.
     05 ERR-COM-ALLOC-RETRY         PIC  9(6) VALUE 501005.
     05 ERR-COM-CONN-ERROR          PIC  9(6) VALUE 501006.
     05 ERR-COM-SEND-ERROR          PIC  9(6) VALUE 501008.
     05 ERR-COM-RECV-RESP-ERR       PIC  9(6) VALUE 501009.
     05 ERR-COM-RESP-TYPE           PIC  9(6) VALUE 501010.
     05 ERR-COM-RESP-MSN            PIC  9(6) VALUE 501011.
     05 ERR-COM-RESP-FATAL          PIC  9(6) VALUE 501012.
     05 ERR-COM-MSG-ERROR           PIC  9(6) VALUE 501013.
     05 ERR-COM-BIG-INDIAN          PIC  9(6) VALUE 501014.
     05 ERR-COM-TSH-ERROR           PIC  9(6) VALUE 501015.
     05 ERR-COM-CCSID-ERROR         PIC  9(6) VALUE 501016.
     05 ERR-COM-MSH-ERROR           PIC  9(6) VALUE 501017.
     05 ERR-COM-MQX-ERROR           PIC  9(6) VALUE 501018.
     05 ERR-COM-INIT-ERROR          PIC  9(6) VALUE 501019.
     05 ERR-COM-FAP-ERROR           PIC  9(6) VALUE 501020.
```

```
05 ERR-COM-MSG-SIZE             PIC  9(6) VALUE 501021.
05 ERR-COM-WRAP-ERROR           PIC  9(6) VALUE 501022.
05 ERR-COM-MCP-DOWN             PIC  9(6) VALUE 501023.
05 ERR-COM-DOWN                 PIC  9(6) VALUE 501024.
05 ERR-COM-NOT-FOUND            PIC  9(6) VALUE 501025.
05 ERR-COM-ERROR                PIC  9(6) VALUE 501026.
05 ERR-COM-BUSY                 PIC  9(6) VALUE 501027.
05 ERR-COM-RESYNC-ERROR         PIC  9(6) VALUE 501028.
05 ERR-COM-STATUS-ERROR         PIC  9(6) VALUE 501029.
05 ERR-COM-LENGTH-ERROR         PIC  9(6) VALUE 501030.
05 ERR-COM-MSG-PER-BATCH        PIC  9(6) VALUE 501031.
05 ERR-COM-MAX-TRANSM-SIZE      PIC  9(6) VALUE 501032.
05 ERR-COM-RESET-MSN            PIC  9(6) VALUE 501050.

05 ERR-INT-LINK-ERROR           PIC  9(6) VALUE 400000.
05 ERR-INT-LINK-COM-SIZE        PIC  9(6) VALUE 400001.
05 ERR-INT-LINK-COM-DATA        PIC  9(6) VALUE 400002.
05 ERR-INT-RETURN-ERROR         PIC  9(6) VALUE 400003.
05 ERR-INT-MOVE-ERROR           PIC  9(6) VALUE 400010.
05 ERR-INT-STRUC-MISSING        PIC  9(6) VALUE 402000.
05 ERR-INT-STRUC-ERROR          PIC  9(6) VALUE 402090.

05 ERR-LOGIC-NOT-SUPPORTED      PIC  9(6) VALUE 300000.
05 ERR-LOGIC-STARTED-WRONG      PIC  9(6) VALUE 300010.
05 ERR-LOGIC-REPEATED-FAILURE   PIC  9(6) VALUE 300020.
05 ERR-LOGIC-LOCKS-EXCEEDED     PIC  9(6) VALUE 300030.
05 ERR-LOGIC-MISSING-RECORD     PIC  9(6) VALUE 301000.
05 ERR-LOGIC-RECORD-DUPLICATED  PIC  9(6) VALUE 301010.
05 ERR-LOGIC-Q-CKP-MISSING      PIC  9(6) VALUE 309010.

05 ERR-PROC-SYSTEM-STOPPED      PIC  9(6) VALUE 100000.
05 ERR-PROC-SYSTEM-ACTIVE       PIC  9(6) VALUE 100010.
05 ERR-PROC-SYS-START-NOQDR     PIC  9(6) VALUE 100011.
05 ERR-PROC-SYS-START-MAXQDR    PIC  9(6) VALUE 100012.
05 ERR-PROC-SYS-START-MAXCOM    PIC  9(6) VALUE 100013.
05 ERR-PROC-SYS-START-NOSYS     PIC  9(6) VALUE 100090.
05 ERR-PROC-Q-EXCEEDED-DEPTH    PIC  9(6) VALUE 101000.
05 ERR-PROC-Q-CONCURRENT-UPD    PIC  9(6) VALUE 101010.
05 ERR-PROC-Q-NOTFOUND          PIC  9(6) VALUE 101015.
05 ERR-PROC-Q-STOPPED           PIC  9(6) VALUE 101090.
05 ERR-PROC-Q-DISABLED          PIC  9(6) VALUE 101091.
05 ERR-PROC-QSN-LIMIT-REACHED   PIC  9(6) VALUE 102090.
05 ERR-PROC-FILE-SPACE-PUT      PIC  9(6) VALUE 102091.
05 ERR-PROC-FILE-SPACE          PIC  9(6) VALUE 102092.
05 ERR-PROC-DUAL-Q-ERROR        PIC  9(6) VALUE 104021.
05 ERR-PROC-DUAL-Q-FILE         PIC  9(6) VALUE 104022.
05 ERR-PROC-DUAL-Q-LOGIC        PIC  9(6) VALUE 104023.
05 ERR-PROC-TRIGGER-ERROR       PIC  9(6) VALUE 105090.
05 ERR-PROC-TRIGGER-DATA        PIC  9(6) VALUE 105091.
05 ERR-PROC-NOT-AUTHORIZED      PIC  9(6) VALUE 109000.

05 ERR-WARN-SYS-STARTED-W-ERR   PIC  9(6) VALUE 010000.
05 ERR-WARN-SYS-STARTED-W-FILER PIC  9(6) VALUE 010001.
05 ERR-WARN-SYS-STARTED-W-COMER PIC  9(6) VALUE 010002.
05 ERR-WARN-SYS-STARTED-W-CHANG PIC  9(6) VALUE 010003.

05 ERR-WARN-COM-CONNECT         PIC  9(6) VALUE 005000.
05 ERR-WARN-COM-OPENED          PIC  9(6) VALUE 005001.
05 ERR-WARN-COM-QUEUE-OPENED    PIC  9(6) VALUE 005002.
05 ERR-WARN-COM-LU62-CONNECT    PIC  9(6) VALUE 005003.
05 ERR-WARN-COM-RECEIVER-ALLOC  PIC  9(6) VALUE 005004.
05 ERR-WARN-COM-QUEUE-EMPTY     PIC  9(6) VALUE 005005.
05 ERR-WARN-COM-QUEUE-CLOSED    PIC  9(6) VALUE 005006.
05 ERR-WARN-COM-DISC            PIC  9(6) VALUE 005007.
05 ERR-WARN-COM-SHUT            PIC  9(6) VALUE 005008.
05 ERR-WARN-COM-SHUT-SENT       PIC  9(6) VALUE 005009.
```

```
     05 ERR-FUNCTION-STARTED         PIC  9(6) VALUE 000100.
     05 ERR-FUNCTION-DONE            PIC  9(6) VALUE 001000.
     05 ERR-FUNCTION-NOT-DONE        PIC  9(6) VALUE 001090.

     05 ERR-WARN-SYS-STARTED         PIC  9(6) VALUE 000000.

     05  SYNCH-MSN-ERROR              PIC 9(6)    VALUE 3.
     05  SYNCH-MSG-DUP                PIC 9(6)    VALUE 4.
     05  LU62-FREE-ERROR              PIC 9(6)    VALUE 10.
     05  LU62-EIB-ERROR               PIC 9(6)    VALUE 11.
     05  LU62-STAT-ERROR              PIC 9(6)    VALUE 12.
     05  LU62-ALLOC-ERROR             PIC 9(6)    VALUE 13.
     05  LU62-ALLOC-RETRY-ERROR       PIC 9(6)    VALUE 14.
     05  LU62-CONN-ERROR              PIC 9(6)    VALUE 15.
     05  LU62-SEND-ERROR              PIC 9(6)    VALUE 16.
     05  LU62-RECV-RESP-ERROR         PIC 9(6)    VALUE 17.
     05  INVLD-RESP-TYPE              PIC 9(6)    VALUE 23.
     05  INVLD-RESP-MSN               PIC 9(6)    VALUE 24.
     05  FATAL-RESP-TYPE              PIC 9(6)    VALUE 25.
     05  RECOVERABLE-RESP-TYPE        PIC 9(6)    VALUE 26.
     05  PARSER-MSN-ERROR             PIC 9(6)    VALUE 29.
     05  PARSER-TYPE-ERROR            PIC 9(6)    VALUE 30.
     05  PARSER-PDM-ERROR             PIC 9(6)    VALUE 31.
     05  PARSER-SID-ERROR             PIC 9(6)    VALUE 32.
     05  PARSER-PN-ERROR              PIC 9(6)    VALUE 33.
     05  PARSER-KEY-ERROR             PIC 9(6)    VALUE 34.
     05  PARSER-APID-ERROR            PIC 9(6)    VALUE 35.
     05  PARSER-ORG-DT-ERROR          PIC 9(6)    VALUE 38.
     05  PARSER-ORIG-MSN-ERROR        PIC 9(6)    VALUE 39.
     05  PARSER-BODY-ERROR            PIC 9(6)    VALUE 40.
     05  PARSER-STATUS-ERROR          PIC 9(6)    VALUE 41.
     05  PARSER-LENGTH-ERROR          PIC 9(6)    VALUE 42.
     05  MCCONN-ERROR                 PIC 9(6)    VALUE 51.
     05  MQOPEN-ERROR                 PIC 9(6)    VALUE 52.
     05  MQGET-ERROR                  PIC 9(6)    VALUE 53.
     05  MQPUT-ERROR                  PIC 9(6)    VALUE 54.
     05  MQPT1-ERROR                  PIC 9(6)    VALUE 55.
     05  MQCLOSE-ERROR                PIC 9(6)    VALUE 56.
     05  MQDISC-ERROR                 PIC 9(6)    VALUE 57.
     05  QM-OTHER-ERROR               PIC 9(6)    VALUE 60.
     05  RECV-RETURN-LON-STATUS       PIC 9(6)    VALUE 80.
     05  RECV-RETURN-LON-TYPE         PIC 9(6)    VALUE 81.
     05  SIDRC-RETURN-MLP-FORMAT      PIC 9(6)    VALUE 91.

*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
* ENVIRONMENT
*-----------------------------------------------------------------*
 01  WS-ENVIR-INFO.
*     COPY   MQICENV.
*-----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQICENV   ***          - BEGIN - *
*-----------------------------------------------------------------*
*   ENVIRONMENT VALUE            - SYSTEM   (ENV)                 *
*-----------------------------------------------------------------*

     02 ENV-DEFINITION.
     03 ENV-DATA-FOR-SYSTEM.
        05 ENV-PRODUCT-INSTALLED       PIC  X(4) VALUE 'MQM '.
          88  ENV-PRODUCT-MQM         VALUE 'MQM '.

        05 ENV-PRODUCT-RUNTIME         PIC  X(4) VALUE 'MQM '.
          88  ENV-PRODUCT-RT-MQM      VALUE 'MQM '.

        05 ENV-LANG-INFO.
           10 ENV-LANGUAGE-FILE-CODE   PIC  99 VALUE 01.
```

```
      10 ENV-LANGUAGE            PIC  X(24)
                                VALUE 'ENGLISH'.
   05 ENV-DATE-FORMAT            PIC  99 VALUE 01.
     88 ENV-DATE-MMDDYY                 VALUE 01.
     88 ENV-DATE-YYMMDD                 VALUE 02.
     88 ENV-DATE-YYDDMM                 VALUE 03.
     88 ENV-DATE-YYDDD                  VALUE 04.
     88 ENV-DATE-DDMMYY                 VALUE 05.


03 ENV-DATA-FOR-TRAN.

   05 ENV-MASTER-TERMINAL-TRAN.
      10 ENV-MT-MASTER-TASK-ID    PIC  X(4) VALUE 'MQMT'.
      10 ENV-MT-CONFIG-TASK-ID    PIC  X(4) VALUE 'MQMC'.
      10 ENV-MT-MONITOR-TASK-ID   PIC  X(4) VALUE 'MQMM'.
      10 ENV-MT-OPER-TASK-ID      PIC  X(4) VALUE 'MQMO'.
      10 ENV-MT-DISP-TASK-ID      PIC  X(4) VALUE 'MQBQ'.
      10 ENV-MT-QUEUE-TASK-ID     PIC  X(4) VALUE 'MQMQ'.
      10 ENV-MT-QUEUEI-TASK-ID    PIC  X(4) VALUE 'MQDQ'.
      10 ENV-MT-COM-TASK-ID       PIC  X(4) VALUE 'MQMH'.
      10 ENV-MT-COMI-TASK-ID      PIC  X(4) VALUE 'MQDH'.
      10 ENV-MT-SYS-TASK-ID       PIC  X(4) VALUE 'MQMS'.
      10 ENV-MT-SYSI-TASK-ID      PIC  X(4) VALUE 'MQDS'.
      10 ENV-MT-MONQ-TASK-ID      PIC  X(4) VALUE 'MQQM'.
      10 ENV-MT-MONC-TASK-ID      PIC  X(4) VALUE 'MQCM'.
      10 ENV-MT-SS-TASK-ID        PIC  X(4) VALUE 'MQMA'.
      10 ENV-MT-SC-TASK-ID        PIC  X(4) VALUE 'MQMB'.
      10 ENV-MT-SI-TASK-ID        PIC  X(4) VALUE 'MQMI'.
      10 ENV-MT-SR-TASK-ID        PIC  X(4) VALUE 'MQMR'.
      10 ENV-MT-SD-TASK-ID        PIC  X(4) VALUE 'MQMD'.
      10 FILLER                   PIC  X(4) VALUE SPACES.
      10 FILLER                   PIC  X(4) VALUE SPACES.
      10 FILLER                   PIC  X(4) VALUE SPACES.

   05 ENV-INTERNAL-ITEMS-TRAN.
      10 ENV-II-MONITOR           PIC  X(4) VALUE 'MQSM'.
      10 ENV-II-M-RECOVERY        PIC  X(4) VALUE 'MQSR'.
      10 ENV-II-Q-RECOVERY        PIC  X(4) VALUE 'MQSQ'.
      10 ENV-II-START-STOP        PIC  X(4) VALUE 'MQSS'.
      10 ENV-II-TRAN-AIP2         PIC  X(4) VALUE 'MQ02'.
      10 ENV-II-TRAN-COM-CHECKP   PIC  X(4) VALUE 'MQCP'.
      10 ENV-II-TRAN-QUE-DELETE   PIC  X(4) VALUE 'MQQD'.
      10 ENV-II-TRAN-QUE-DEL-ALL  PIC  X(4) VALUE 'MQQA'.
      10 FILLER                   PIC  X(4) VALUE SPACES.
      10 FILLER                   PIC  X(4) VALUE SPACES.
      10 FILLER                   PIC  X(4) VALUE SPACES.


03 ENV-DATA-FOR-PROGRAMS.

   05 ENV-MASTER-TERMINAL-PROGRAMS.
      10 ENV-MT-MASTER-PROGRAM    PIC  X(8) VALUE 'MQPMTP'.
      10 ENV-MT-CONFIG-PROGRAM    PIC  X(8) VALUE 'MQPMCFG'.
      10 ENV-MT-MONITOR-PROGRAM   PIC  X(8) VALUE 'MQPMMON'.
      10 ENV-MT-OPER-PROGRAM      PIC  X(8) VALUE 'MQPMOPR'.
      10 ENV-MT-DISP-PROGRAM      PIC  X(8) VALUE 'MQPDISP'.
      10 ENV-MT-QUEUE-PROGRAM     PIC  X(8) VALUE 'MQPMQUE'.
      10 ENV-MT-QUEUEI-PROGRAM    PIC  X(8) VALUE 'MQPMQUE'.
      10 ENV-MT-COM-PROGRAM       PIC  X(8) VALUE 'MQPMCOM'.
      10 ENV-MT-COMI-PROGRAM      PIC  X(8) VALUE 'MQPMCOM'.
      10 ENV-MT-SYS-PROGRAM       PIC  X(8) VALUE 'MQPMSYS'.
      10 ENV-MT-SYSI-PROGRAM      PIC  X(8) VALUE 'MQPMSYS'.
      10 ENV-MT-MONQ-PROGRAM      PIC  X(8) VALUE 'MQPMMOQ'.
      10 ENV-MT-MONC-PROGRAM      PIC  X(8) VALUE 'MQPMMOC'.
      10 ENV-MT-SS-PROGRAM        PIC  X(8) VALUE 'MQPMSS'.
      10 ENV-MT-SC-PROGRAM        PIC  X(8) VALUE 'MQPMSC'.
```

```
           10 ENV-MT-SI-PROGRAM        PIC  X(8) VALUE 'MQPMSI'.
           10 ENV-MT-SR-PROGRAM        PIC  X(8) VALUE 'MQPMMSN'.
           10 ENV-MT-SD-PROGRAM        PIC  X(8) VALUE 'MQPMDEL'.
           10 ENV-MT-CMD-PROGRAM       PIC  X(8) VALUE 'MQPCMD'.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.

        05 ENV-INTERNAL-ITEMS-PROGRAMS.
           10 ENV-II-LINK-ERROR        PIC  X(8) VALUE 'MQPERR  '.
           10 ENV-II-LINK-EIB1         PIC  X(8) VALUE 'MQPEIB1 '.
           10 ENV-II-LINK-AIP0         PIC  X(8) VALUE 'MQPAIP0 '.
           10 ENV-II-LINK-AIP1         PIC  X(8) VALUE 'MQPAIP1 '.
           10 ENV-II-LINK-AIP2         PIC  X(8) VALUE 'MQPAIP2 '.

           10 ENV-II-LINK-ECHO         PIC  X(8) VALUE 'MQPECHO '.
           10 ENV-II-LINK-FINDQ        PIC  X(8) VALUE 'MQPFINDQ'.
           10 ENV-II-LINK-QUE1         PIC  X(8) VALUE 'MQPQUE1 '.
           10 ENV-II-LINK-QUE2         PIC  X(8) VALUE 'MQPQUE2 '.
           10 ENV-II-LINK-INIT1        PIC  X(8) VALUE 'MQPINIT1'.
           10 ENV-II-LINK-INIT2        PIC  X(8) VALUE 'MQPINIT2'.
           10 ENV-II-LINK-SSQ          PIC  X(8) VALUE 'MQPSSQ  '.
           10 ENV-II-LINK-SCHK         PIC  X(8) VALUE 'MQPSCHK '.
           10 ENV-II-LINK-SREC         PIC  X(8) VALUE 'MQPSREC '.
           10 ENV-II-LINK-QRECOVERY    PIC  X(8) VALUE 'MQPQREC '.
           10 ENV-II-LINK-SENDER       PIC  X(8) VALUE 'MQPSEND '.
           10 ENV-II-LINK-RECIEVER     PIC  X(8) VALUE 'MQPRECV '.
           10 ENV-II-LINK-COM-CHECKP   PIC  X(8) VALUE 'MQPCCKPT'.
           10 ENV-II-LINK-QUE-DELETE   PIC  X(8) VALUE 'MQPQDEL'.
           10 ENV-II-LINK-SET-MAP      PIC  X(8) VALUE 'MQPSMAP'.
           10 ENV-II-LINK-LU21         PIC  X(8) VALUE 'MQPLU21'.
           10 ENV-II-LINK-LU33         PIC  X(8) VALUE 'MQPLU33'.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.

     03 ENV-DATA-FOR-MAPS.

        05 ENV-MASTER-TERMINAL-MAPS.
           10 ENV-MT-MASTER-MAPSCREEN  PIC  X(8) VALUE 'MQMMTP'.
           10 ENV-MT-CONFIG-MAPSCREEN  PIC  X(8) VALUE 'MQMMCFG'.
           10 ENV-MT-MONITOR-MAPSCREEN PIC  X(8) VALUE 'MQMMMON'.
           10 ENV-MT-OPER-MAPSCREEN    PIC  X(8) VALUE 'MQMMOPR'.
           10 ENV-MT-DISP-MAPSCREEN    PIC  X(8) VALUE 'MQMDISP'.
           10 ENV-MT-QUEUE-MAPSCREEN   PIC  X(8) VALUE 'MQMMQUE'.
           10 ENV-MT-QUEUEI-MAPSCREEN  PIC  X(8) VALUE 'MQMMQUE'.
           10 ENV-MT-COM-MAPSCREEN     PIC  X(8) VALUE 'MQMMCOM'.
           10 ENV-MT-COMI-MAPSCREEN    PIC  X(8) VALUE 'MQMMCOM'.
           10 ENV-MT-SYS-MAPSCREEN     PIC  X(8) VALUE 'MQMMSYS'.
           10 ENV-MT-SYSI-MAPSCREEN    PIC  X(8) VALUE 'MQMMSYS'.
           10 ENV-MT-MONQ-MAPSCREEN    PIC  X(8) VALUE 'MQMMMOQ'.
           10 ENV-MT-MONC-MAPSCREEN    PIC  X(8) VALUE 'MQMMMOC'.
           10 ENV-MT-SS-MAPSCREEN      PIC  X(8) VALUE 'MQMMSS'.
           10 ENV-MT-SC-MAPSCREEN      PIC  X(8) VALUE 'MQMMSC'.
           10 ENV-MT-SI-MAPSCREEN      PIC  X(8) VALUE 'MQMMSI'.
           10 ENV-MT-SR-MAPSCREEN      PIC  X(8) VALUE 'MQMMMSN'.
           10 ENV-MT-SD-MAPSCREEN      PIC  X(8) VALUE 'MQMMDEL'.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.
           10 FILLER                   PIC  X(8) VALUE SPACES.


     03 ENV-DATA-FOR-CONSTANTS.

        05 ENV-CONFIG-DDNAME           PIC  X(8) VALUE 'MQFCNFG'.
        05 ENV-SYSTEM-NUMBER           PIC  9(4) VALUE 1.
        05 ENV-MASTER-TERMINAL-CONS.
           10 ENV-MT-TITLE             PIC  X(40) VALUE
```

```
        '   IBM MQSeries for VSE/ESA Version 2 '.

        05 ENV-INTERNAL-ITEMS-CONS.
           10 ENV-II-ERROR-TD          PIC  X(4) VALUE 'MQER'.
           10 ENV-II-ERROR-CSMT        PIC  X(4) VALUE 'CSMT'.
           10 ENV-II-SYSTEM-ANCHOR     PIC  X(8) VALUE 'MQTAQM'.
           10 ENV-II-SYSTEM-PREFIX     PIC  X(4) VALUE 'MQI '.
           10 ENV-II-DUMPCODE          PIC  X(4) VALUE 'MQ??'.
           10 ENV-II-ENQ-INIT1         PIC  X(8) VALUE 'MQPINIT1'.
           10 ENV-II-SYSTEM-ENVIR      PIC  X(8) VALUE 'MQTENV  '.
           10 ENV-IT-UN-INIT-MSG       PIC  X(80) VALUE
      'MQ900000: MQSERIES VSE ENVIRONMENT not initialized.'.
           10 FILLER                   PIC  X(80) VALUE SPACES.

*-----------------------------------------------------------------*
* - END   -      ***  COPYBOOK: MQICENV   ***          - END   - *
*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
* USER PROCESS DEFINITION
*-----------------------------------------------------------------*
 01  WS-PROC.
*     COPY CMQTMV.
************************************************************************
**                                                                    **
**  FILE NAME:        CMQTMV                                          **
**                                                                    **
**  DESCRIPTIVE NAME: COBOL copy file for MQTM structure              **
**                                                                    **
**  VERSION 1.3.0                                                   **
**                                                                    **
**  FUNCTION:         This file declares the MQTM structure,        **
**                    which forms part of the IBM Message           **
**                    Queue Interface (MQI).                        **
**                                                                    **
************************************************************************

**   MQTM structure
  10 MQTM.
**    Structure identifier
   15 MQTM-STRUCID      PIC X(4) VALUE 'TM  '.
**    Structure version number
   15 MQTM-VERSION      PIC S9(9) BINARY   VALUE 1.
**    Name of triggered queue
   15 MQTM-QNAME.
     25  MQI-PROC-LOCAL-QUEUE-NAME PIC  X(48) VALUE SPACE.

**    Name of process object
   15 MQTM-PROCESSNAME  PIC X(48) VALUE SPACES.

**    Trigger data
   15 MQTM-TRIGGERDATA  PIC X(64) VALUE SPACES.
   15 MQTM-TRIGGERDATA-RED  REDEFINES MQTM-TRIGGERDATA.
     25  MQI-PROC-TRANS-ID         PIC  X(4).
     25  MQI-PROC-PROGRAM-ID       PIC  X(8).

     25  MQI-PROC-TRIGGER-EVENT    PIC  X.
       88 MQI-PROC-TRIGGER-FIRST       VALUE 'F'.
       88 MQI-PROC-TRIGGER-EVERY       VALUE 'E'.

**    Application type
   15 MQTM-APPLTYPE     PIC S9(9) BINARY   VALUE 0.
**    Application identifier
   15 MQTM-APPLID       PIC X(256) VALUE SPACES.
**    Environment data
   15 MQTM-ENVDATA      PIC X(128) VALUE SPACES.
**    User data
```

```
   15 MQTM-USERDATA     PIC X(128) VALUE SPACES.
   15 MQTM-USERDATA-RED  REDEFINES  MQTM-USERDATA.
     25  MQI-PROC-CHANNEL-NAME     PIC  X(20).

*----------------------------------------------------------------*

*----------------------------------------------------------------*
 01  MQI-VALUES.
*----------------------------------------------------------------*
*     COPY    CMQV.
******************************************************************
**                                                              **
**  FILE NAME:        CMQV                                      **
**                                                              **
**  DESCRIPTIVE NAME: COBOL copy file for MQI constants         **
**                                                              **
**  VERSION 1.3.0                                             **
**                                                              **
**  FUNCTION:         This file declares the constants          **
**                    which form part of the IBM Message        **
**                    Queue Interface (MQI).                    **
**                                                              **
******************************************************************

******************************************************************
**  Values Related to MQDLH Structure                           **
******************************************************************
**   Structure Identifier
  10 MQDLH-STRUC-ID PIC X(4) VALUE 'DLH '.

**   Structure Version Number
  10 MQDLH-VERSION-1 PIC S9(9) BINARY  VALUE 1.


******************************************************************
**  Values Related to MQGMO Structure                           **
******************************************************************

**   Structure Identifier
  10 MQGMO-STRUC-ID PIC X(4) VALUE 'GMO '.

**   Structure Version Number
  10 MQGMO-VERSION-1 PIC S9(9) BINARY  VALUE 1.

**   Get-Message Options
  10 MQGMO-WAIT                    PIC S9(9) BINARY  VALUE 1.
  10 MQGMO-NO-WAIT                 PIC S9(9) BINARY  VALUE 0.
  10 MQGMO-BROWSE-FIRST            PIC S9(9) BINARY  VALUE 16.
  10 MQGMO-BROWSE-NEXT             PIC S9(9) BINARY  VALUE 32.
  10 MQGMO-ACCEPT-TRUNCATED-MSG    PIC S9(9) BINARY  VALUE 64.
  10 MQGMO-SET-SIGNAL              PIC S9(9) BINARY  VALUE 8.
  10 MQGMO-SYNCPOINT               PIC S9(9) BINARY  VALUE 2.
  10 MQGMO-NO-SYNCPOINT            PIC S9(9) BINARY  VALUE 4.
  10 MQGMO-MSG-UNDER-CURSOR        PIC S9(9) BINARY  VALUE 256.
  10 MQGMO-LOCK                    PIC S9(9) BINARY  VALUE 512.
  10 MQGMO-UNLOCK                  PIC S9(9) BINARY  VALUE 1024.

**   Wait Interval
  10 MQWI-UNLIMITED PIC S9(9) BINARY  VALUE -1.


******************************************************************
**  Values Related to MQMD Structure                            **
******************************************************************

**   Structure Identifier
  10 MQMD-STRUC-ID PIC X(4) VALUE 'MD  '.
```

```
**   Structure Version Number
  10 MQMD-VERSION-1 PIC S9(9) BINARY  VALUE 1.

**   Report Options
  10 MQRO-NONE PIC S9(9) BINARY  VALUE 0.

**   Message Types
  10 MQMT-REQUEST  PIC S9(9) BINARY  VALUE 1.
  10 MQMT-REPLY    PIC S9(9) BINARY  VALUE 2.
  10 MQMT-DATAGRAM PIC S9(9) BINARY  VALUE 8.
  10 MQMT-REPORT   PIC S9(9) BINARY  VALUE 4.

**   Expiry Value
  10 MQEI-UNLIMITED PIC S9(9) BINARY  VALUE -1.

**   Feedback Values
  10 MQFB-NONE         PIC S9(9) BINARY  VALUE 0.
  10 MQFB-QUIT         PIC S9(9) BINARY  VALUE 256.
  10 MQFB-SYSTEM-FIRST PIC S9(9) BINARY  VALUE 1.
  10 MQFB-SYSTEM-LAST  PIC S9(9) BINARY  VALUE 65535.
  10 MQFB-APPL-FIRST   PIC S9(9) BINARY  VALUE 65536.
  10 MQFB-APPL-LAST    PIC S9(9) BINARY  VALUE 999999999.

* format
  10 MQFMT-NONE                  PIC  X(8) VALUE SPACES.
  10 MQFMT-DEAD-LETTER-Q-HEADER  PIC  X(8) VALUE 'MQDLQH'.
  10 MQFMT-TRIGGER               PIC  X(8) VALUE 'MQTRIG'.
  10 MQFMT-XMIT-Q-HEADER         PIC  X(8) VALUE 'MQXMIT'.

**   Encoding Value
  10 MQENC-NATIVE PIC S9(9) BINARY  VALUE 785.

**   Encoding Masks
  10 MQENC-INTEGER-MASK  PIC S9(9) BINARY  VALUE 15.
  10 MQENC-DECIMAL-MASK  PIC S9(9) BINARY  VALUE 240.
  10 MQENC-FLOAT-MASK    PIC S9(9) BINARY  VALUE 3840.
  10 MQENC-RESERVED-MASK PIC S9(9) BINARY  VALUE -4096.

**   Encodings for Binary Integers
  10 MQENC-INTEGER-UNDEFINED PIC S9(9) BINARY  VALUE 0.
  10 MQENC-INTEGER-NORMAL    PIC S9(9) BINARY  VALUE 1.
  10 MQENC-INTEGER-REVERSED  PIC S9(9) BINARY  VALUE 2.

**   Encodings for Packed-Decimal Integers
  10 MQENC-DECIMAL-UNDEFINED PIC S9(9) BINARY  VALUE 0.
  10 MQENC-DECIMAL-NORMAL    PIC S9(9) BINARY  VALUE 16.
  10 MQENC-DECIMAL-REVERSED  PIC S9(9) BINARY  VALUE 32.

**   Encodings for Floating-Point Numbers
  10 MQENC-FLOAT-UNDEFINED     PIC S9(9) BINARY  VALUE 0.
  10 MQENC-FLOAT-IEEE-NORMAL   PIC S9(9) BINARY  VALUE 256.
  10 MQENC-FLOAT-IEEE-REVERSED PIC S9(9) BINARY  VALUE 512.
  10 MQENC-FLOAT-S390          PIC S9(9) BINARY  VALUE 768.

**   Coded Character-Set Identifier
  10 MQCCSI-Q-MGR PIC S9(9) BINARY  VALUE 0.

**   Persistence Values
  10 MQPER-PERSISTENT           PIC S9(9) BINARY  VALUE 1.
  10 MQPER-PERSISTENCE-AS-Q-DEF PIC S9(9) BINARY  VALUE 2.

**   Message Id Value
  10 MQMI-NONE PIC X(24) VALUE LOW-VALUES.

**   Correllation Id Value
  10 MQCI-NONE PIC X(24) VALUE LOW-VALUES.
```

```
      ****************************************************************
      **  Values Related to MQOD Structure                          **
      ****************************************************************

      **   Structure Identifier
        10 MQOD-STRUC-ID PIC X(4) VALUE 'OD  '.

      **   Structure Version Number
        10 MQOD-VERSION-1 PIC S9(9) BINARY  VALUE 1.

      **   Object Types
        10 MQOT-Q PIC S9(9) BINARY  VALUE 1.


      ****************************************************************
      **  Values Related to MQPMO Structure                         **
      ****************************************************************

      **   Structure Identifier
        10 MQPMO-STRUC-ID PIC X(4) VALUE 'PMO '.

      **   Structure Version Number
        10 MQPMO-VERSION-1 PIC S9(9) BINARY  VALUE 1.

      **   Put-Message Options
        10 MQPMO-SYNCPOINT            PIC S9(9) BINARY  VALUE 2.
        10 MQPMO-NO-SYNCPOINT         PIC S9(9) BINARY  VALUE 4.

      ****************************************************************
      **  Values Related to MQTM Structure                          **
      ****************************************************************

      **   Structure Identifier
        10 MQTM-STRUC-ID PIC X(4) VALUE 'TM  '.

      **   Structure Version Number
        10 MQTM-VERSION-1 PIC S9(9) BINARY  VALUE 1.


      ****************************************************************
      **  Values Related to MQCLOSE Call                            **
      ****************************************************************

      **   Close Options
        10 MQCO-NONE PIC S9(9) BINARY  VALUE 0.


      ****************************************************************
      **  Values Related to MQINQ Call                              **
      ****************************************************************

      **   Character-Attribute Selectors
        10 MQCA-BASE-Q-NAME       PIC S9(9) BINARY  VALUE 2002.
        10 MQCA-CREATION-DATE     PIC S9(9) BINARY  VALUE 2004.
        10 MQCA-CREATION-TIME     PIC S9(9) BINARY  VALUE 2005.
        10 MQCA-FIRST             PIC S9(9) BINARY  VALUE 2001.
        10 MQCA-INITIATION-Q-NAME PIC S9(9) BINARY  VALUE 2008.
        10 MQCA-LAST              PIC S9(9) BINARY  VALUE 4000.
        10 MQCA-PROCESS-NAME      PIC S9(9) BINARY  VALUE 2012.
        10 MQCA-Q-DESC            PIC S9(9) BINARY  VALUE 2013.
        10 MQCA-Q-NAME            PIC S9(9) BINARY  VALUE 2016.
        10 MQCA-REMOTE-Q-MGR-NAME PIC S9(9) BINARY  VALUE 2017.
        10 MQCA-REMOTE-Q-NAME     PIC S9(9) BINARY  VALUE 2018.

      **   Integer-Attribute Selectors
```

```
  10 MQIA-CURRENT-Q-DEPTH   PIC S9(9) BINARY  VALUE 3.
  10 MQIA-DEF-PERSISTENCE   PIC S9(9) BINARY  VALUE 5.
  10 MQIA-DEFINITION-TYPE   PIC S9(9) BINARY  VALUE 7.
  10 MQIA-FIRST             PIC S9(9) BINARY  VALUE 1.
  10 MQIA-INHIBIT-GET       PIC S9(9) BINARY  VALUE 9.
  10 MQIA-INHIBIT-PUT       PIC S9(9) BINARY  VALUE 10.
  10 MQIA-LAST              PIC S9(9) BINARY  VALUE 2000.
  10 MQIA-MAX-MSG-LENGTH    PIC S9(9) BINARY  VALUE 13.
  10 MQIA-MAX-Q-DEPTH       PIC S9(9) BINARY  VALUE 15.
  10 MQIA-OPEN-INPUT-COUNT  PIC S9(9) BINARY  VALUE 17.
  10 MQIA-OPEN-OUTPUT-COUNT PIC S9(9) BINARY  VALUE 18.
  10 MQIA-Q-TYPE            PIC S9(9) BINARY  VALUE 20.
  10 MQIA-SHAREABILITY      PIC S9(9) BINARY  VALUE 23.
  10 MQIA-TRIGGER-CONTROL   PIC S9(9) BINARY  VALUE 24.
  10 MQIA-TRIGGER-TYPE      PIC S9(9) BINARY  VALUE 28.
  10 MQIA-USAGE             PIC S9(9) BINARY  VALUE 12.

**   Integer Attribute Value Denoting 'Not Applicable'
  10 MQIAV-NOT-APPLICABLE PIC S9(9) BINARY  VALUE -1.


****************************************************************
**  Values Related to MQOPEN Call                             **
****************************************************************

**   Open Options
  10 MQOO-INPUT-SHARED    PIC S9(9) BINARY  VALUE 2.
  10 MQOO-INPUT-EXCLUSIVE PIC S9(9) BINARY  VALUE 4.
  10 MQOO-BROWSE          PIC S9(9) BINARY  VALUE 8.
  10 MQOO-OUTPUT          PIC S9(9) BINARY  VALUE 16.
  10 MQOO-INQUIRE         PIC S9(9) BINARY  VALUE 32.


****************************************************************
**  Values Related to All Calls                               **
****************************************************************

**   String Lengths
  10 MQ-CREATION-DATE-LENGTH     PIC S9(9) BINARY  VALUE 12.
  10 MQ-CREATION-TIME-LENGTH     PIC S9(9) BINARY  VALUE 8.
  10 MQ-PROCESS-APPL-ID-LENGTH   PIC S9(9) BINARY  VALUE 256.
  10 MQ-PROCESS-DESC-LENGTH      PIC S9(9) BINARY  VALUE 64.
  10 MQ-PROCESS-ENV-DATA-LENGTH  PIC S9(9) BINARY  VALUE 128.
  10 MQ-PROCESS-NAME-LENGTH      PIC S9(9) BINARY  VALUE 48.
  10 MQ-PROCESS-USER-DATA-LENGTH PIC S9(9) BINARY  VALUE 128.
  10 MQ-Q-DESC-LENGTH            PIC S9(9) BINARY  VALUE 64.
  10 MQ-Q-NAME-LENGTH            PIC S9(9) BINARY  VALUE 48.
  10 MQ-Q-MGR-DESC-LENGTH        PIC S9(9) BINARY  VALUE 64.
  10 MQ-Q-MGR-NAME-LENGTH        PIC S9(9) BINARY  VALUE 48.
  10 MQ-TRIGGER-DATA-LENGTH      PIC S9(9) BINARY  VALUE 64.

**   Completion Codes
  10 MQCC-OK      PIC S9(9) BINARY  VALUE 0.
  10 MQCC-WARNING PIC S9(9) BINARY  VALUE 1.
  10 MQCC-FAILED  PIC S9(9) BINARY  VALUE 2.

**   Reason Codes
  10 MQRC-NONE                       PIC S9(9) BINARY  VALUE 0.
  10 MQRC-ACCESS-RESTRICTED          PIC S9(9) BINARY  VALUE 2000.
  10 MQRC-ALIAS-BASE-Q-TYPE-ERROR    PIC S9(9) BINARY  VALUE 2001.
  10 MQRC-ALREADY-CONNECTED          PIC S9(9) BINARY  VALUE 2002.
  10 MQRC-BUFFER-ERROR               PIC S9(9) BINARY  VALUE 2004.
  10 MQRC-BUFFER-LENGTH-ERROR        PIC S9(9) BINARY  VALUE 2005.
  10 MQRC-CHAR-ATTR-LENGTH-ERROR     PIC S9(9) BINARY  VALUE 2006.
  10 MQRC-CHAR-ATTRS-ERROR           PIC S9(9) BINARY  VALUE 2007.
  10 MQRC-CHAR-ATTRS-TOO-SHORT       PIC S9(9) BINARY  VALUE 2008.
  10 MQRC-CONNECTION-BROKEN          PIC S9(9) BINARY  VALUE 2009.
```

```
10 MQRC-DATA-LENGTH-ERROR         PIC S9(9) BINARY  VALUE 2010.
10 MQRC-EXPIRY-ERROR              PIC S9(9) BINARY  VALUE 2013.
10 MQRC-FEEDBACK-ERROR            PIC S9(9) BINARY  VALUE 2014.
10 MQRC-GET-INHIBITED             PIC S9(9) BINARY  VALUE 2016.
10 MQRC-HANDLE-NOT-AVAILABLE      PIC S9(9) BINARY  VALUE 2017.
10 MQRC-HCONN-ERROR               PIC S9(9) BINARY  VALUE 2018.
10 MQRC-HOBJ-ERROR                PIC S9(9) BINARY  VALUE 2019.
10 MQRC-INT-ATTR-COUNT-ERROR      PIC S9(9) BINARY  VALUE 2021.
10 MQRC-INT-ATTR-COUNT-TOO-SMALL  PIC S9(9) BINARY  VALUE 2022.
10 MQRC-INT-ATTRS-ARRAY-ERROR     PIC S9(9) BINARY  VALUE 2023.
10 MQRC-MAX-CONNS-LIMIT-REACHED   PIC S9(9) BINARY  VALUE 2025.
10 MQRC-MD-ERROR                  PIC S9(9) BINARY  VALUE 2026.
10 MQRC-MISSING-REPLY-TO-Q        PIC S9(9) BINARY  VALUE 2027.
10 MQRC-MSG-TYPE-ERROR            PIC S9(9) BINARY  VALUE 2029.
10 MQRC-MSG-TOO-BIG-FOR-Q         PIC S9(9) BINARY  VALUE 2030.
10 MQRC-NO-MSG-AVAILABLE          PIC S9(9) BINARY  VALUE 2033.
10 MQRC-NO-MSG-UNDER-CURSOR       PIC S9(9) BINARY  VALUE 2034.
10 MQRC-NOT-AUTHORIZED            PIC S9(9) BINARY  VALUE 2035.
10 MQRC-NOT-OPEN-FOR-BROWSE       PIC S9(9) BINARY  VALUE 2036.
10 MQRC-NOT-OPEN-FOR-INPUT        PIC S9(9) BINARY  VALUE 2037.
10 MQRC-NOT-OPEN-FOR-INQUIRE      PIC S9(9) BINARY  VALUE 2038.
10 MQRC-NOT-OPEN-FOR-OUTPUT       PIC S9(9) BINARY  VALUE 2039.
10 MQRC-OBJECT-CHANGED            PIC S9(9) BINARY  VALUE 2041.
10 MQRC-OBJECT-IN-USE             PIC S9(9) BINARY  VALUE 2042.
10 MQRC-OBJECT-TYPE-ERROR         PIC S9(9) BINARY  VALUE 2043.
10 MQRC-OD-ERROR                  PIC S9(9) BINARY  VALUE 2044.
10 MQRC-OPTION-NOT-VALID-FOR-TYPE PIC S9(9) BINARY  VALUE 2045.
10 MQRC-OPTIONS-ERROR             PIC S9(9) BINARY  VALUE 2046.
10 MQRC-PERSISTENCE-ERROR         PIC S9(9) BINARY  VALUE 2047.
10 MQRC-PRIORITY-EXCEEDS-MAXIMUM  PIC S9(9) BINARY  VALUE 2049.
10 MQRC-PRIORITY-ERROR            PIC S9(9) BINARY  VALUE 2050.
10 MQRC-PUT-INHIBITED             PIC S9(9) BINARY  VALUE 2051.
10 MQRC-Q-FULL                    PIC S9(9) BINARY  VALUE 2053.
10 MQRC-Q-SPACE-NOT-AVAILABLE     PIC S9(9) BINARY  VALUE 2056.
10 MQRC-Q-MGR-NAME-ERROR          PIC S9(9) BINARY  VALUE 2058.
10 MQRC-Q-MGR-NOT-AVAILABLE       PIC S9(9) BINARY  VALUE 2059.
10 MQRC-REPORT-OPTIONS-ERROR      PIC S9(9) BINARY  VALUE 2061.
10 MQRC-SECURITY-ERROR            PIC S9(9) BINARY  VALUE 2063.
10 MQRC-SELECTOR-COUNT-ERROR      PIC S9(9) BINARY  VALUE 2065.
10 MQRC-SELECTOR-LIMIT-EXCEEDED   PIC S9(9) BINARY  VALUE 2066.
10 MQRC-SELECTOR-ERROR            PIC S9(9) BINARY  VALUE 2067.
10 MQRC-SELECTOR-NOT-FOR-TYPE     PIC S9(9) BINARY  VALUE 2068.
10 MQRC-SIGNAL-OUTSTANDING        PIC S9(9) BINARY  VALUE 2069.
10 MQRC-SIGNAL-REQUEST-ACCEPTED   PIC S9(9) BINARY  VALUE 2070.
10 MQRC-STORAGE-NOT-AVAILABLE     PIC S9(9) BINARY  VALUE 2071.
10 MQRC-SYNCPOINT-NOT-AVAILABLE   PIC S9(9) BINARY  VALUE 2072.
10 MQRC-TRUNCATED-MSG-ACCEPTED    PIC S9(9) BINARY  VALUE 2079.
10 MQRC-TRUNCATED-MSG-FAILED      PIC S9(9) BINARY  VALUE 2080.
10 MQRC-UNEXPECTED-CONNECT-ERROR  PIC S9(9) BINARY  VALUE 2081.
10 MQRC-UNKNOWN-ALIAS-BASE-Q      PIC S9(9) BINARY  VALUE 2082.
10 MQRC-UNKNOWN-OBJECT-NAME       PIC S9(9) BINARY  VALUE 2085.
10 MQRC-UNKNOWN-OBJECT-Q-MGR      PIC S9(9) BINARY  VALUE 2086.
10 MQRC-UNKNOWN-REMOTE-Q-MGR      PIC S9(9) BINARY  VALUE 2087.
10 MQRC-WAIT-INTERVAL-ERROR       PIC S9(9) BINARY  VALUE 2090.
10 MQRC-XMIT-Q-TYPE-ERROR         PIC S9(9) BINARY  VALUE 2091.
10 MQRC-XMIT-Q-USAGE-ERROR        PIC S9(9) BINARY  VALUE 2092.
10 MQRC-PMO-ERROR                 PIC S9(9) BINARY  VALUE 2173.
10 MQRC-GMO-ERROR                 PIC S9(9) BINARY  VALUE 2186.

10 MQRC-UNEXPECTED-ERROR          PIC S9(9) BINARY  VALUE 2195.
10 MQRC-MSG-ID-ERROR              PIC S9(9) BINARY  VALUE 2206.
10 MQRC-CORREL-ID-ERROR           PIC S9(9) BINARY  VALUE 2207.

10 MQRC-FILE-SYSTEM-ERROR         PIC S9(9) BINARY  VALUE 2208.
10 MQRC-NO-MSG-LOCKED             PIC S9(9) BINARY  VALUE 2209.
```

```
*****************************************************************
**  Values Related to Queue Attributes                         **
*****************************************************************

**   Queue Types
  10 MQQT-LOCAL  PIC S9(9) BINARY  VALUE 1.
  10 MQQT-ALIAS  PIC S9(9) BINARY  VALUE 3.
  10 MQQT-REMOTE PIC S9(9) BINARY  VALUE 6.

**   Queue Definition Types
  10 MQQDT-PREDEFINED PIC S9(9) BINARY  VALUE 1.

**   Inhibit Get
  10 MQQA-GET-INHIBITED PIC S9(9) BINARY  VALUE 1.
  10 MQQA-GET-ALLOWED   PIC S9(9) BINARY  VALUE 0.

**   Inhibit Put
  10 MQQA-PUT-INHIBITED PIC S9(9) BINARY  VALUE 1.
  10 MQQA-PUT-ALLOWED   PIC S9(9) BINARY  VALUE 0.

**   Queue Shareability
  10 MQQA-SHAREABLE     PIC S9(9) BINARY  VALUE 1.
  10 MQQA-NOT-SHAREABLE PIC S9(9) BINARY  VALUE 0.

**   Message Delivery Sequence
  10 MQMDS-FIFO PIC S9(9) BINARY  VALUE 1.

**   Trigger Control
  10 MQTC-OFF PIC S9(9) BINARY  VALUE 0.
  10 MQTC-ON  PIC S9(9) BINARY  VALUE 1.

**   Trigger Types
  10 MQTT-NONE  PIC S9(9) BINARY  VALUE 0.
  10 MQTT-FIRST PIC S9(9) BINARY  VALUE 1.
  10 MQTT-EVERY PIC S9(9) BINARY  VALUE 2.

**   Queue Usage
  10 MQUS-NORMAL       PIC S9(9) BINARY  VALUE 0.
  10 MQUS-TRANSMISSION PIC S9(9) BINARY  VALUE 1.


*****************************************************************
**  Values Related to Process-Definition Attributes            **
*****************************************************************

**   Application Type
  10 MQAT-USER-FIRST PIC S9(9) BINARY  VALUE 65536.
  10 MQAT-USER-LAST  PIC S9(9) BINARY  VALUE 999999999.


*
  10 MQAT-OS2        PIC S9(9) BINARY VALUE 4.
  10 MQAT-DOS        PIC S9(9) BINARY VALUE 5.
  10 MQAT-AIX        PIC S9(9) BINARY VALUE 6.
  10 MQAT-OS400      PIC S9(9) BINARY VALUE 8.
  10 MQAT-WINDOWS    PIC S9(9) BINARY VALUE 9.
  10 MQAT-CICS-VSE   PIC S9(9) BINARY VALUE 10.
  10 MQAT-VMS        PIC S9(9) BINARY VALUE 12.
  10 MQAT-GUARDIAN   PIC S9(9) BINARY VALUE 13.
  10 MQAT-VOS        PIC S9(9) BINARY VALUE 14.


*****************************************************************
**  Values Related to Queue-Manager Attributes                 **
*****************************************************************

**   Syncpoint Availability
```

```
  10 MQSP-AVAILABLE PIC S9(9) BINARY  VALUE 1.

*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
* API
*-----------------------------------------------------------------*
*     COPY   MQIAIP1.
*-----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQIAIP1   ***          - BEGIN - *
*-----------------------------------------------------------------*
*   9/ 1/93   REV:                                                *
*-----------------------------------------------------------------*
*   APPL. INTERFACE PARM FOR SSI STUBS                            *
*-----------------------------------------------------------------*

     05 API-CALL-PARM.
        10 API-FUNCTION                 PIC X(4).
           88 API-CONNECT               VALUE 'CONN', 'CONI'
                                              'MCCO'.
           88 API-CONNECT-VIA-APPL      VALUE 'CONN', 'CONI'.
           88 API-CONNECT-VIA-INTERFACE VALUE 'CONI'.
           88 API-MCP-CONNECT           VALUE 'MCCO'.
           88 API-OPEN                  VALUE 'OPEN'.
           88 API-PUT                   VALUE 'PUT '.
           88 API-INQ                   VALUE 'INQ '.
           88 API-GET                   VALUE 'GET '.
           88 API-GET-QSN               VALUE 'GETQ'.
           88 API-CLOSE                 VALUE 'CLOS'.
           88 API-DISCONNECT            VALUE 'DISC'.

        10 API-RETURN-CODE-INFO.
           15  API-CCODE-ADDR        USAGE POINTER.
           15  API-RCODE-ADDR        USAGE POINTER.

        10 API-VARIABLE-PARM-INFO.
           15  API-HCONN-ADDR        USAGE POINTER.
           15  API-HOBJ-ADDR         USAGE POINTER.
           15  API-PARM-NUM          PIC S9(4) COMP.
           15  FILLER                PIC  XX.
           15  API-PARM-ADDR-LIST.
             20  API-PARM-ADDR  OCCURS 50 TIMES
                                    USAGE POINTER.

*-----------------------------------------------------------------*
* - END -        ***  COPYBOOK: MQIAIP1   ***            - END - *
*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
*     COPY     MQIENQ.
*-----------------------------------------------------------------*
*  "MQIENQ"                                                       *
*-----------------------------------------------------------------*
*      ENQ/DEQ  DEFINITIONS        FOR QUEUEING/COM. HANDLERS    *
*-----------------------------------------------------------------*
 01  ENQ-INFO.

*--GLOBAL ENVIRONMENT TS QUEUE ID
     05  ENQ-ENVIR-TS-INFO.
         10  ENQ-ENVIR-TS-ITEM     PIC S9(4) COMP VALUE +1.
         10  ENQ-ENVIR-TS-SIZE     PIC S9(4) COMP VALUE ZERO.
         10  ENQ-ENVIR-TS-QID      PIC  X(8) VALUE 'MQSERIES'.

*--ENQ KEY FOR LOCKING
     05  ENQ-RECORD.
         10  ENQ-QSN               PIC  9(10) VALUE ZERO.
         10  ENQ-OBJ-NAME          PIC  X(48) VALUE SPACES.
```

```
     05  QSN-BUSY-FLAG            PIC  X   VALUE SPACE.
        88 QSN-BUSY                        VALUE 'Y'.
        88 QSN-BUSY-OK                     VALUE 'N'.

*--QUE RECORD RIB  KEY
     05  QUEUE-KEY.
         10  QUEUE-KEY-OBJ        PIC  X(48) VALUE SPACES.
         10  QUEUE-KEY-QSN        PIC S9(8) COMP VALUE ZERO.

*--DRQ TS QUEUE ID
     05  ENQ-RT-QUEUE-ID.
         10  ENQ-RT-CONSTANT      PIC  X(3) VALUE 'MQT'.
         10  ENQ-RT-TYPE          PIC  X   VALUE 'Q'.
         10  ENQ-RT-HHHH          PIC  9999 VALUE ZERO.
         10  ENQ-RT-ITEM          PIC  9999 VALUE ZERO.

*--DRQ WAIT REQID
     05  ENQ-RT-REQID-ID.
         10  ENQ-RT-R-CONSTANT    PIC  X(3) VALUE 'MQT'.
         10  ENQ-RT-R-TYPE        PIC  X   VALUE 'Q'.
         10  ENQ-RT-R-HHHH        PIC  9999 COMP VALUE ZERO.
         10  ENQ-RT-R-ITEM        PIC  9999 COMP VALUE ZERO.

*--DELETE QUEUE  TS QUEUE ID
     05  ENQ-DQ-QUEUE-ID.
         10  ENQ-DQ-CONSTANT      PIC  X(3) VALUE 'MQT'.
         10  ENQ-DQ-TYPE          PIC  X   VALUE 'D'.
         10  ENQ-DQ-HHHH          PIC  9999 VALUE ZERO.
         10  ENQ-DQ-ITEM          PIC  9999 COMP VALUE ZERO.


*--ENQ FOR COMMUNICATION HANDLERS - SENDERS
     05  ENQ-COMH-ID.
         10  ENQ-COMH-CONSTANT       PIC  X(3) VALUE 'MQT'.
         10  ENQ-COMH-ENTRY          PIC  9(5) VALUE ZERO.

*----------------------------------------------------------------*


*----------------------------------------------------------------*
*--OPEN    PARM
 01  WS-Q-NAME-AREA.
*    COPY  CMQODV.
************************************************************************
**                                                                    **
**  FILE NAME:        CMQODV                                          **
**                                                                    **
**  DESCRIPTIVE NAME: COBOL copy file for MQOD structure              **
**                                                                    **
**  VERSION 1.3.0                                                   **
**                                                                    **
**  FUNCTION:         This file declares the MQOD structure,        **
**                    which forms part of the IBM Message           **
**                    Queue Interface (MQI).                        **
**                                                                    **
************************************************************************

**   MQOD structure
  10 MQOD.
**    Structure identifier
   15 MQOD-STRUCID         PIC X(4) VALUE 'OD  '.
**    Structure version number
   15 MQOD-VERSION         PIC S9(9) BINARY   VALUE 1.
**    Object type
   15 MQOD-OBJECTTYPE      PIC S9(9) BINARY   VALUE 1.
**    Object name
```

```
   15 MQOD-OBJECTNAME      PIC X(48) VALUE SPACES.
**    Object queue manager name
   15 MQOD-OBJECTQMGRNAME  PIC X(48) VALUE SPACES.
**    Dynamic queue name
   15 MQOD-DYNAMICQNAME    PIC X(48) VALUE '*'.
**    Alternate user identifier
   15 MQOD-ALTERNATEUSERID PIC X(12) VALUE SPACES.


*--INQ
 01  MQI-SELECTOR-COUNT.
     05  WS-SELECTOR-COUNT          PIC S9(8) COMP.
 01  MQI-SELECTOR.
     05  WS-SELECTOR                PIC XXXX.

 01  MQI-IN-ATTR-COUNT.
     05  WS-IN-ATTR-COUNT           PIC S9(8) COMP.
 01  MQI-IN-ATTR.
     05  WS-IN-ATTR                 PIC XXXX.

 01  MQI-CHAR-ATTR-LENGTH.
     05  WS-CHAR-ATTR-LENGTH        PIC S9(8) COMP.
 01  MQI-CHAR-ATTR.
     05  WS-CHAR-ATTR               PIC XXXX.


*--PUT/GET PARM
 01  WS-MSG-DESCRIPTOR.
*    COPY   CMQMDV.

****************************************************************
**                                                            **
**  FILE NAME:        CMQMDV                                  **
**                                                            **
**  DESCRIPTIVE NAME: COBOL copy file for MQMD structure      **
**                                                            **
**  VERSION 1.3.0                                           **
**                                                            **
**  FUNCTION:         This file declares the MQMD structure,  **
**                    which forms part of the IBM Message     **
**                    Queue Interface (MQI).                  **
**                                                            **
****************************************************************

**   MQMD structure
  10 MQMD.
**    Structure identifier
   15 MQMD-STRUCID           PIC X(4) VALUE 'MD  '.
**    Structure version number
   15 MQMD-VERSION           PIC S9(9) BINARY  VALUE 1.
**    Reserved
   15 MQMD-REPORT            PIC S9(9) BINARY  VALUE 0.
**    Message type
   15 MQMD-MSGTYPE           PIC S9(9) BINARY  VALUE 8.
**    Reserved
   15 MQMD-EXPIRY            PIC S9(9) BINARY  VALUE -1.
**    Feedback code
   15 MQMD-FEEDBACK          PIC S9(9) BINARY  VALUE 0.
**    Data encoding
   15 MQMD-ENCODING          PIC S9(9) BINARY  VALUE 785.
**    Coded character set identifier
   15 MQMD-CODEDCHARSETID    PIC S9(9) BINARY  VALUE 0.
**    Format name
   15 MQMD-FORMAT            PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-PRIORITY          PIC S9(9) BINARY  VALUE 0.
**    Message persistence
```

```
   15 MQMD-PERSISTENCE       PIC S9(9) BINARY  VALUE 2.
**    Message identifier
   15 MQMD-MSGID             PIC X(24) VALUE LOW-VALUES.
**    Correlation identifier
   15 MQMD-CORRELID          PIC X(24) VALUE LOW-VALUES.
**    Reserved
   15 MQMD-BACKOUTCOUNT      PIC S9(9) BINARY  VALUE 0.
**    Name of reply queue
   15 MQMD-REPLYTOQ          PIC X(48) VALUE SPACES.
**    Name of reply queue manager
   15 MQMD-REPLYTOQMGR       PIC X(48) VALUE SPACES.
**    Reserved
   15 MQMD-USERIDENTIFIER    PIC X(12) VALUE SPACES.
**    Reserved
   15 MQMD-ACCOUNTINGTOKEN   PIC X(32) VALUE LOW-VALUES.
**    Reserved
   15 MQMD-APPLIDENTITYDATA  PIC X(32) VALUE SPACES.
**    Reserved
   15 MQMD-PUTAPPLTYPE       PIC S9(9) BINARY  VALUE 0.
**    Reserved
   15 MQMD-PUTAPPLNAME       PIC X(28) VALUE SPACES.
**    Reserved
   15 MQMD-PUTDATE           PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-PUTTIME           PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-APPLORIGINDATA    PIC X(4) VALUE SPACES.

 01  WS-PUT-OPTIONS.
*    COPY   CMQPMOV.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQPMOV                                  **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQPMO structure      **
**                                                             **
**  VERSION 1.3.0                                              **
**                                                             **
**  FUNCTION:         This file declares the MQPMO structure,  **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************

**   MQPMO structure
  10 MQPMO.
**    Structure identifier
   15 MQPMO-STRUCID          PIC X(4) VALUE 'PMO '.
**    Structure version number
   15 MQPMO-VERSION          PIC S9(9) BINARY   VALUE 1.
**    Reserved
   15 MQPMO-OPTIONS          PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-TIMEOUT          PIC S9(9) BINARY   VALUE -1.
**    Reserved
   15 MQPMO-CONTEXT          PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-KNOWNDESTCOUNT   PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-UNKNOWNDESTCOUNT  PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQPMO-INVALIDDESTCOUNT  PIC S9(9) BINARY   VALUE 0.
**    Resolved name of destination queue
   15 MQPMO-RESOLVEDQNAME     PIC X(48) VALUE SPACES.
**    Resolved name of destination queue manager
   15 MQPMO-RESOLVEDQMGRNAME  PIC X(48) VALUE SPACES.
```

```
 01  WS-GET-OPTIONS.
*    COPY    CMQGMOV.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQGMOV                                  **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQGMO structure      **
**                                                             **
**  VERSION 1.3.0                                              **
**                                                             **
**  FUNCTION:         This file declares the MQGMO structure,  **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************

**   MQGMO structure
  10 MQGMO.
**    Structure identifier
   15 MQGMO-STRUCID        PIC X(4) VALUE 'GMO '.
**    Structure version number
   15 MQGMO-VERSION        PIC S9(9) BINARY   VALUE 1.
**    Options
   15 MQGMO-OPTIONS        PIC S9(9) BINARY   VALUE 0.
**    Wait interval
   15 MQGMO-WAITINTERVAL   PIC S9(9) BINARY   VALUE 0.
**    Signal
   15 MQGMO-SIGNAL1        PIC S9(9) BINARY   VALUE 0.
**    Reserved
   15 MQGMO-SIGNAL2        PIC S9(9) BINARY   VALUE 0.
**    Resolved name of destination queue
   15 MQGMO-RESOLVEDQNAME  PIC X(48) VALUE SPACES.

*-------------------------------------------------------------*
* COMMON PARMS
 01  WS-PARMS.
     05  WS-HCONN-VALUE             USAGE POINTER.
     05  WS-HOBJ-VALUE              USAGE POINTER.
     05  WS-PUT-HOBJ-VALUE          USAGE POINTER.
     05  WS-CCODE-VALUE             PIC S9(8) COMP.
     05  WS-RCODE-VALUE             PIC S9(8) COMP.
     05  WS-QM-NAME-CONNECT         PIC  X(48).
     05  WS-Q-OPEN-OPTIONS-VALUE    PIC S9(8) COMP.
     05  WS-DATA-LENGTH-USER        PIC S9(8) COMP.
     05  WS-BUFFER-LENGTH           PIC S9(8) COMP.

     05  WS-BUFFER-AREA.
         10 FILLER                  PIC X(8000).


*    COPY  MQWEOWS.
*-------------------------------------------------------------*
*Debugging eyecatcher information for end of WORKING-STORAGE.
*-------------------------------------------------------------*
 01  FILLER                      PIC X(16)  VALUE
      '================'.
 01  WS-RWS-PROGRAM-NAME2        PIC X(8).
 01  FILLER                      PIC X(16) VALUE
      ' Version V2.1.0'.
*-------------------------------------------------------------*
*-------------------------------------------------------------*
 LINKAGE SECTION.
*-------------------------------------------------------------*
 01  DFHCOMMAREA.
     05  FILLER                    PIC X(1000).
```

```
*-------------------------------------------------------------*

*-------------------------------------------------------------*
 PROCEDURE DIVISION.
*-------------------------------------------------------------*

 0000-MAIN-LINE.
     MOVE 'MQPECHO'  TO WS-RWS-PROGRAM-NAME1
                        WS-RWS-PROGRAM-NAME2.
     MOVE WHEN-COMPILED TO WS-RWS-WHEN-COMPILED.

*--INITIALIZE
     PERFORM 1000-INITIALIZE
        THRU 1000-EXIT.

*--CONNECT AND OPEN GET QUEUE
     PERFORM 2000-CONNECT.
     PERFORM 3100-GET-OPEN.

*-------------------------------------------------------------*
     SET  WS-MORE-DATA   TO TRUE.
     PERFORM
          UNTIL (WS-NOMORE-DATA)

*--GET MESSAGE
          PERFORM 3500-GET-MESSAGES

     END-PERFORM.


*--CLOSE AND DISC
     PERFORM 3900-GET-CLOSE.
     PERFORM 5000-DISCONNECT.

*-------------------------------------------------------------*
 0000-RETURN.
     EXEC CICS RETURN
          END-EXEC.

     GOBACK.

*-------------------------------------------------------------*
 1000-INITIALIZE.
*-------------------------------------------------------------*
* PURPOSE: SETUP DATA AREAS
*-------------------------------------------------------------*

*--GET ENVIRONMENT INFO
     PERFORM 1015-GET-ENVRIR-RECORD
        THRU 1015-GET-ENVRIR-EXIT.

*--SET UP ERROR AREA
     PERFORM   1050-SET-ERROR-INFO.
*
*--CHECK IF QUEUE PRESENT
     IF EIBCALEN   <   LENGTH OF MQTM
      THEN
          GO TO  0000-RETURN.

*--MOVE QUEUE NAME
     MOVE  DFHCOMMAREA    TO  MQTM.

*-------------------------------------------------------------*
 1000-EXIT.
     EXIT.

*-------------------------------------------------------------*
```

```
 1015-GET-ENVRIR-RECORD.
*-------------------------------------------------------------*
* PURPOSE: READ ENVIRONMENT RECORD
*-------------------------------------------------------------*
*--SET HANDLE
     EXEC CICS HANDLE CONDITION
                     QIDERR  (9900-NO-ENVIR-SETUP)
                     ITEMERR (9900-NO-ENVIR-SETUP)
     END-EXEC.


*--READ  ANCHOR FOR QM
     MOVE  LENGTH OF ENV-DEFINITION   TO ENQ-ENVIR-TS-SIZE.
     EXEC CICS READQ TS
         QUEUE  (ENQ-ENVIR-TS-QID)
         INTO   (ENV-DEFINITION)
         LENGTH (ENQ-ENVIR-TS-SIZE)
         ITEM   (ENQ-ENVIR-TS-ITEM)
     END-EXEC.

*--CHECK IF GOOD SIZE
     IF  LENGTH OF ENV-DEFINITION
                 NOT EQUAL  ENQ-ENVIR-TS-SIZE
         GO TO  9900-NO-ENVIR-SETUP
     END-IF.

*-------------------------------------------------------------*
 1015-GET-ENVRIR-EXIT.
     EXIT.

*-------------------------------------------------------------*
 1050-SET-ERROR-INFO.
*-------------------------------------------------------------*
* PURPOSE: SET DEFAULT ERROR INFO
*-------------------------------------------------------------*
*--SET CSMT DATE AND TIME
     EXEC CICS ASKTIME
               ABSTIME(WS-ABSTIME)
     END-EXEC.

     MOVE EIBTIME           TO WS-UNPACK-TIME-9.
     MOVE WS-TIME-HH        TO WS-FORMAT-TIME-HH
     MOVE WS-TIME-MM        TO WS-FORMAT-TIME-MM.
     MOVE WS-TIME-SS        TO WS-FORMAT-TIME-SS.

     EXEC CICS FORMATTIME
               ABSTIME (WS-ABSTIME)
               MMDDYY  (WS-FORMATTED-DATE)
               DATESEP ('/')
     END-EXEC.
*
     EXEC  CICS FORMATTIME
                ABSTIME(WS-ABSTIME)
                YYMMDD (WS-DATE-YYMMDD)
     END-EXEC.

*-- --SET CENTURY
     IF WS-DATE-YY >  50
         MOVE  19                 TO  WS-DATE-CC
     ELSE
         MOVE  20                 TO  WS-DATE-CC
     END-IF.

*--SET COMMON ERROR INFO
     MOVE  ZERO         TO  ERR-CODE.
     MOVE  ENV-II-LINK-ECHO  TO  ERR-PROGRAM.
```

```
*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
 2000-CONNECT.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT
*-----------------------------------------------------------------*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
     MOVE MQCC-OK                       TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE                     TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'MQCONN'  USING  WS-QM-NAME
                           WS-HCONN-VALUE
                           WS-CCODE-VALUE
                           WS-RCODE-VALUE.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        GO TO  9900-ERR-DISPLAY
     END-IF.
*
*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
 3100-GET-OPEN.
*-----------------------------------------------------------------*
* PURPOSE: OPEN
*-----------------------------------------------------------------*
*--MQOPEN   QUEUE TO QM
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-INPUT-SHARED TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE SPACES            TO   MQOD-OBJECTQMGRNAME.
     MOVE MQI-PROC-LOCAL-QUEUE-NAME
                            TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK           TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE         TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE  TO NULL.
     CALL 'MQOPEN'  USING  WS-HCONN-VALUE
                           WS-Q-NAME-AREA
                           WS-Q-OPEN-OPTIONS-VALUE
                           WS-HOBJ-VALUE
                           WS-CCODE-VALUE
                           WS-RCODE-VALUE.

*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        GO TO  9900-ERR-DISPLAY
     END-IF.

*
*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
 3500-GET-MESSAGES.
*-----------------------------------------------------------------*
*--MQGET TO QUEUE TO QM
     MOVE  'GET'     TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     MOVE LOW-VALUES TO  MQMD-MSGID
                         MQMD-CORRELID.
     MOVE LENGTH OF WS-BUFFER-AREA  TO   WS-BUFFER-LENGTH.
     MOVE MQGMO-ACCEPT-TRUNCATED-MSG
                               TO  MQGMO-OPTIONS.

     CALL 'MQGET'   USING  WS-HCONN-VALUE
```

```
                          WS-HOBJ-VALUE
                          WS-MSG-DESCRIPTOR
                          WS-GET-OPTIONS
                          WS-BUFFER-LENGTH
                          WS-BUFFER-AREA
                          WS-DATA-LENGTH
                          WS-CCODE-VALUE
                          WS-RCODE-VALUE.
*
*
     IF  (WS-CCODE-VALUE   NOT EQUAL  ZERO)
     AND (WS-RCODE-VALUE   NOT EQUAL MQRC-TRUNCATED-MSG-ACCEPTED)
         IF WS-RCODE-VALUE EQUAL MQRC-NO-MSG-AVAILABLE
            SET WS-NOMORE-DATA  TO TRUE
         ELSE
            GO TO  9900-ERR-DISPLAY
         END-IF
     END-IF.

*--SEND QUEUE RECORDS
     IF WS-MORE-DATA
*-- --FIRST CHECK IF ANY REPLY
         PERFORM 4000-PUT1-MESSAGES
            THRU 4000-EXIT
     END-IF.


*--SYNCPOINT
     EXEC CICS SYNCPOINT
     END-EXEC.
*
*
*----------------------------------------------------------------*

*----------------------------------------------------------------*
 3900-GET-CLOSE.
*----------------------------------------------------------------*
* PURPOSE: CLOSE
*----------------------------------------------------------------*
*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQCLOSE' USING  WS-HCONN-VALUE
                           WS-HOBJ-VALUE
                           WS-Q-OPEN-OPTIONS-VALUE
                           WS-CCODE-VALUE
                           WS-RCODE-VALUE.
*
*
     IF  WS-CCODE-VALUE   NOT EQUAL  ZERO
         GO TO  9900-ERR-DISPLAY
     END-IF.


*----------------------------------------------------------------*

*----------------------------------------------------------------*
 4000-PUT1-MESSAGES.
*----------------------------------------------------------------*
* PURPOSE: PUT1
*----------------------------------------------------------------*
*--MQPUT1   QUEUE TO QM
     MOVE  'PUT1'    TO   WS-FUNCTION.
     MOVE MQOO-OUTPUT    TO     WS-Q-OPEN-OPTIONS-VALUE.
     IF MQMD-REPLYTOQMGR    EQUAL SPACES OR LOW-VALUES
```

```
           MOVE SPACES   TO   MQOD-OBJECTQMGRNAME
      ELSE
           MOVE MQMD-REPLYTOQMGR
                         TO   MQOD-OBJECTQMGRNAME
     END-IF.
*
*--IF NOT REPLY QUEUE - SET DEFAULT
     IF MQMD-REPLYTOQ     EQUAL SPACES OR LOW-VALUES
           MOVE WS-R-Q-NAME
                         TO   MQOD-OBJECTNAME
      ELSE
           MOVE MQMD-REPLYTOQ
                         TO   MQOD-OBJECTNAME
     END-IF.

*
     MOVE MQCC-OK        TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE      TO   WS-RCODE-VALUE.
     MOVE WS-DATA-LENGTH-USER   TO WS-BUFFER-LENGTH.
     CALL 'MQPUT1'  USING  WS-HCONN-VALUE
                           WS-Q-NAME-AREA
                           WS-MSG-DESCRIPTOR
                           WS-PUT-OPTIONS
                           WS-BUFFER-LENGTH
                           WS-BUFFER-AREA
                           WS-CCODE-VALUE
                           WS-RCODE-VALUE.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        GO TO  9900-ERR-DISPLAY
     END-IF.

*-----------------------------------------------------------------*
 4000-EXIT.
     EXIT.

*-----------------------------------------------------------------*
 5000-DISCONNECT.
*-----------------------------------------------------------------*
* PURPOSE:  DISCON
*-----------------------------------------------------------------*
*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'MQDISC'  USING
                           WS-HCONN-VALUE
                           WS-CCODE-VALUE
                           WS-RCODE-VALUE.
*

*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
 9900-ERR-DISPLAY.
*-----------------------------------------------------------------*
*--ERROR IN "MQ" VERB
*
     MOVE ERR-INT-RETURN-ERROR        TO ERR-CODE.
     MOVE MQI-PROC-LOCAL-QUEUE-NAME   TO ERR-QUEUE.
     PERFORM 9999-CONVERT-ERROR-INFO.

*--WRITE ERROR
     PERFORM 9999-ERROR-WRITE.
*
*--ALWAYS DISCONNECT (NOTE NO ERROR CHECKING IN DISCONNECT)
```

```
      *--SYNCPOINT - ROLLBACK
           EXEC CICS SYNCPOINT
                ROLLBACK
           END-EXEC.
      *
           PERFORM      5000-DISCONNECT.
           GO TO  0000-RETURN.
      *

      *---------------------------------------------------------------*
       9900-CICS-PGMIDERR.
      *---------------------------------------------------------------*

      *--SET MESSAGE AND CODE
           MOVE ERR-CICS-PGMIDERR  TO ERR-CODE.

      *--CONVERT ERROR CODE
           PERFORM 9999-CONVERT-ERROR-INFO.

      *--WRITE ERROR
           PERFORM 9999-ERROR-WRITE.

      *--RETURN
           GO TO  0000-RETURN.


      *---------------------------------------------------------------*
      * ERROR PROCESSING
      *---------------------------------------------------------------*
      *    COPY     MQIERRCD.
/INCLUDE MQIERRCD

      *--ADDED CODE FOR ABEND CONDITION

      *--RETURN
           GO TO  0000-RETURN.

       9900-NO-ENVIR-SETUP.
           GO TO  0000-RETURN.
```

Appendix E. Example configuration - MQSeries for VSE/ESA Version 2.1.2

This appendix gives an example of how to set up communication links from MQSeries for VSE/ESA to MQSeries products on the following platforms:

- OS/2
- Windows NT
- AIX®
- HP-UX
- AT&T GIS UNIX (This platform has become NCR UNIX SVR4 MP-RAS, R3.0)
- Sun Solaris
- OS/400®
- MVS/ESA without CICS

It describes the parameters needed for an LU 6.2 and TCP/IP connection. Once the connection is established, you need to define some channels to complete the configuration. This is described in "MQSeries for VSE/ESA configuration" on page 456.

Configuration parameters for an LU 6.2 connection

Table 22 presents a worksheet listing all the parameters needed to set up communication from VSE/ESA to one of the other MQSeries platforms. The worksheet shows examples of the parameters, which have been tested in a working environment, and leaves space for you to fill in your own values. An explanation of the parameter names follows the worksheet. Use the worksheet in this chapter in conjunction with the worksheet in the *MQSeries Intercommunication* book for the platform to which you are connecting.

Configuration worksheet

Use the following worksheet to record the values you will use for this configuration. Where numbers appear in the Reference column they indicate that the value must match that in the appropriate worksheet in the *MQSeries Intercommunication* book. The examples that follow in this chapter refer back to the values in the ID column of this table. The entries in the Parameter Name column are explained in "Explanation of terms" on page 453.

Table 22. Configuration worksheet for VSE/ESA using APPC

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
1	Network ID		**NETID**	
2	Node name		**VSEPU**	
3	Local LU name		**VSELU**	
4	Local Transaction Program name		**MQ01**	**MQ01**
5	LAN destination address		**400074511092**	
Connection to an OS/2 system The values in this section of the table must match those used in the table for OS/2 in the *MQSeries Intercommunication* book, as indicated.				

© Copyright IBM Corp. 1995, 2004

Table 22. Configuration worksheet for VSE/ESA using APPC (continued)

ID	Parameter Name	Reference	Example Used	User Value
6	Connection name		**OS2**	
7	Group name		**EXAMPLE**	
8	Session name		**OS2SESS**	
9	Netname	6	**OS2LU**	
Connection to a Windows NT system The values in this section of the table must match those used in the table for Windows NT in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**WNT**	
7	Group name		**EXAMPLE**	
8	Session name		**WNTSESS**	
9	Netname	5	**WINNTLU**	
Connection to an AIX system The values in this section of the table must match those used in the table for AIX in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**AIX**	
7	Group name		**EXAMPLE**	
8	Session name		**AIXSESS**	
9	Netname	4	**AIXLU**	
Connection to an HP-UX system The values in this section of the table must match those used in the table for HP-UX in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**HPUX**	
7	Group name		**EXAMPLE**	
8	Session name		**HPUXSESS**	
9	Netname	5	**HPUXLU**	
Connection to an AT&T GIS UNIX system The values in this section of the table must match those used in the table for GIS UNIX in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**GIS**	
7	Group name		**EXAMPLE**	
8	Session name		**GISSESS**	
9	Netname	4	**GISLU**	
Connection to a Sun Solaris system The values in this section of the table must match those used in the table for Sun Solaris in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**SOL**	
7	Group name		**EXAMPLE**	
8	Session name		**SOLSESS**	
9	Netname	5	**SOLARLU**	

Table 22. Configuration worksheet for VSE/ESA using APPC (continued)

ID	Parameter Name	Reference	Example Used	User Value
Connection to an AS/400 system The values in this section of the table must match those used in the table for AS/400 in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**AS4**	
7	Group name		**EXAMPLE**	
8	Session name		**AS4SESS**	
9	Netname	3	**AS400LU**	
Connection to an MVS/ESA system without CICS The values in this section of the table must match those used in the table for MVS/ESA in the *MQSeries Intercommunication* book, as indicated.				
6	Connection name		**MVS**	
7	Group name		**EXAMPLE**	
8	Session name		**MVSSESS**	
9	Netname	4	**MVSLU**	

Explanation of terms

1 Network ID
This is the unique ID of the network to which you are connected. Your system administrator will tell you this value.

2 Node name
This is the name of the SSCP which owns the CICS/VSE region.

3 Local LU name
This is the unique VTAM APPLID of this CICS/VSE region.

4 Transaction Program name
MQSeries applications trying to converse with this queue manager will specify a transaction name for the program to be run at the receiving end. This will have been defined on the channel definition at the sender. MQSeries for VSE/ESA uses a name of MQ01.

5 LAN destination address
This is the LAN destination address that your partner nodes will use to communicate with this host. It is usually the address of the 3745 on the same LAN as the partner node.

6 Connection name
This is a 4-character name by which each connection will be individually known in CICS RDO.

7 Group name
You choose your own 8-character name for this value. Your system may already have a group defined for connections to partner nodes. Your system administrator will give you a value to use.

8 Session name
This is an 8-character name by which each session will be individually known. For clarity we use the connection name, concatenated with 'SESS'.

9 Netname

This is the LU name of the MQSeries queue manager on the system with which you are setting up communication.

Establishing an LU 6.2 connection

This example is for a connection to an OS/2 system. The steps are the same whatever platform you are using; change the values as appropriate.

Defining a connection

1. At a CICS command line type:

   CEDA DEF CONN(*connection name*) 6 GROUP(*group name*) 7

   For example:

   CEDA DEF CONN(OS2) GROUP(EXAMPLE)

2. Press Enter to define a connection to CICS.

```
DEF CONN(OS2) GROUP(EXAMPLE)
OVERTYPE TO MODIFY
 CEDA  DEFine
  Connection     : OS2
  Group          : EXAMPLE
  DEscription  ==>
 CONNECTION IDENTIFIERS
  Netname      ==> OS2LU
  INDsys       ==>
 REMOTE ATTRIBUTES
  REMOTESystem ==>
  REMOTEName   ==>
 CONNECTION PROPERTIES
  ACcessmethod ==> Vtam              Vtam | IRc | INdirect | Xm
  Protocol     ==> Appc              Appc | Lu61
  SInglesess   ==> No                No | Yes
  DAtastream   ==> User              User | 3270 | SCs | STrfield | Lms
  RECordformat ==> U                 U | Vb
 OPERATIONAL PROPERTIES
+ AUtoconnect  ==> Yes               No | Yes | All
  I New group EXAMPLE created.

  DEFINE SUCCESSFUL                        TIME:  16.49.30  DATE: 96.054
PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

3. On the panel change the **Netname** field in the CONNECTION IDENTIFIERS section to be the LU name (9) of the target system.
4. In the CONNECTION PROPERTIES section set the **ACcessmethod** field to `Vtam` and the **Protocol** to `Appc`.
5. Press Enter to make the change.

Defining a session

1. At a CICS command line type:

   CEDA DEF SESS(*session name*) 8 GROUP(*group name*) 7

   For example:

   CEDA DEF SESS(OS2SESS) GROUP(EXAMPLE)

2. Press Enter to define a session for the connection.

```
DEF SESS(OS2SESS) GROUP(EXAMPLE)
OVERTYPE TO MODIFY
 CEDA  DEFine
  Sessions     ==> OS2SESS
  Group        ==> EXAMPLE
  DEscription  ==>
 SESSION IDENTIFIERS
  Connection   ==> OS2
  SESSName     ==>
  NETnameq     ==>
  MOdename     ==> #INTER
 SESSION PROPERTIES
  Protocol     ==> Appc              Appc | Lu61
  MAximum      ==> 008 , 004         0-999
  RECEIVEPfx   ==>
  RECEIVECount ==>                   1-999
  SENDPfx      ==>
  SENDCount    ==>                   1-999
  SENDSize     ==> 04096             1-30720
+ RECEIVESize  ==> 04096             1-30720
  S CONNECTION MUST BE SPECIFIED.

                                        TIME:  14.23.19   DATE:  96.054
PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

3. In the SESSION IDENTIFIERS section of the panel specify the Connection name (6) in the **Connection** field and set the **MOdename** to `#INTER`.
4. In the SESSION PROPERTIES section set the **Protocol** to `Appc` and the **MAximum** field to `008 , 004`.

Installing the new group definition

1. At a CICS command line type:

   CEDA INS GROUP(*group name*) 7

2. Press Enter to install the new group definition.

   Note: If this connection group is already in use you may get severe errors reported. If this happens, take the existing connections out of service, retry the above group installation, and then set the connections in service using the following commands:

   a. CEMT I CONN
   b. CEMT S CONN(*) OUTS
   c. CEDA INS GROUP(*group name*)
   d. CEMT S CONN(*) INS

What next?

The LU 6.2 connection is now established. You are ready to complete the configuration. Go to "MQSeries for VSE/ESA configuration" on page 456.

Establishing a TCP/IP connection

TCP/IP connections do not require the configuration of additional profiles as does the LU 6.2 protocol. Instead, MQSeries for VSE/ESA processes the MQSeries listener program during MQSeries startup.

The MQSeries listener program waits for remote TCP/IP connection requests. As these are received, the listener starts the receiver MCA to process the remote connection. When the remote connection is received from a client program, the receiver MCA starts the MQSeries server program.

Note: There is one MQSeries server process for each client connection.

Provided that the MQ Listener is active and TCP/IP is active in a VSE partition, TCP/IP connections can be established.

MQSeries for VSE/ESA configuration

Configuring MQSeries for VSE/ESA involves the following tasks:
- Configuring channels
- Defining a local queue
- Defining a remote queue
- Defining a sender channel
- Defining a receiver channel

Note: MQSeries for VSE/ESA does not understand the format of the MQSeries channel ping command. The only way to verify your MQSeries definitions is to start the channels and put messages onto remote queues.

Configuring channels

Examples are given for connecting MQSeries for VSE/ESA and MQSeries for OS/2 Warp. If you wish connect to another MQSeries platform use the appropriate set of values from the table in place of those for OS/2.

Note: The words in **bold** are user-specified and reflect the names of MQSeries objects used throughout these examples. If you change the names used here, ensure that you also change the other references made to these objects throughout this book. All others are keywords and should be entered as shown.

Refer to the sections “Defining a local queue” on page 460 and “Defining a remote queue” on page 460 for details of how to create queue definitions, and “Defining a SNA LU 6.2 sender channel” on page 461 and “Defining a SNA LU 6.2 receiver channel” on page 462 for details of how to create channels.

Table 23. Configuration worksheet for MQSeries for VSE/ESA

ID	Parameter Name	Reference	Example Used	User Value
Definition for local node				
A	Queue Manager Name		**VSEP**	
B	Local queue name		**VSE.LOCALQ**	
Connection to MQSeries for OS/2 Warp The values in this section of the table must match those used in the worksheet table for OS/2 in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name	A	**OS2**	
D	Remote queue name		**OS2.REMOTEQ**	
E	Queue name at remote system	B	**OS2.LOCALQ**	
F	Transmission queue name		**OS2**	
G	Sender channel name		**VSE.OS2.SNA**	
I	Receiver channel name	G	**OS2.VSE.SNA**	
Connection to MQSeries for Windows NT The values in this section of the table must match those used in the worksheet table for Windows NT in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name	A	**WINNT**	
D	Remote queue name		**WINNT.REMOTEQ**	
E	Queue name at remote system	B	**WINNT.LOCALQ**	
F	Transmission queue name		**WINNT**	
G	Sender channel name		**VSE.WINNT.SNA**	
I	Receiver channel name	G	**WINNT.VSE.SNA**	
Connection to MQSeries for AIX The values in this section of the table must match those used in the worksheet table for AIX in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name		**AIX**	
D	Remote queue name		**AIX.REMOTEQ**	
E	Queue name at remote system	B	**AIX.LOCALQ**	
F	Transmission queue name		**AIX**	
G	Sender channel name		**VSE.AIX.SNA**	
I	Receiver channel name	G	**AIX.VSE.SNA**	
Connection to MQSeries for HP-UX The values in this section of the table must match those used in the worksheet table for HP-UX in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name		**HPUX**	
D	Remote queue name		**HPUX.REMOTEQ**	
E	Queue name at remote system	B	**HPUX.LOCALQ**	
F	Transmission queue name		**HPUX**	
G	Sender channel name		**VSE.HPUX.SNA**	
I	Receiver channel name	G	**HPUX.VSE.SNA**	
Connection to MQSeries for AT&T GIS UNIX The values in this section of the table must match those used in the worksheet table for GIS UNIX in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name		**GIS**	
D	Remote queue name		**GIS.REMOTEQ**	
E	Queue name at remote system	B	**GIS.LOCALQ**	

Table 23. Configuration worksheet for MQSeries for VSE/ESA (continued)

ID	Parameter Name	Reference	Example Used	User Value
F	Transmission queue name		**GIS**	
G	Sender channel name		**VSE.GIS.SNA**	
I	Receiver channel name	G	**GIS.VSE.SNA**	
Connection to MQSeries for Sun Solaris The values in this section of the table must match those used in the worksheet table for Sun Solaris in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name		**SOLARIS**	
D	Remote queue name		**SOLARIS.REMOTEQ**	
E	Queue name at remote system	B	**SOLARIS.LOCALQ**	
F	Transmission queue name		**SOLARIS**	
G	Sender channel name		**VSE.SOLARIS.SNA**	
I	Receiver channel name	G	**SOLARIS.VSE.SNA**	
Connection to MQSeries for AS/400 The values in this section of the table must match those used in the worksheet table for AS/400 in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name		**AS400**	
D	Remote queue name		**AS400.REMOTEQ**	
E	Queue name at remote system	B	**AS400.LOCALQ**	
F	Transmission queue name		**AS400**	
G	Sender channel name		**VSE.AS400.SNA**	
I	Receiver channel name	G	**AS400.VSE.SNA**	
Connection to MQSeries for OS/390 without CICS The values in this section of the table must match those used in the worksheet table for MVS/ESA in the *MQSeries Intercommunication* book, as indicated.				
C	Remote queue manager name		**MVS**	
D	Remote queue name		**MVS.REMOTEQ**	
E	Queue name at remote system	B	**MVS.LOCALQ**	
F	Transmission queue name		**MVS**	
G	Sender channel name		**VSE.MVS.SNA**	
I	Receiver channel name	G	**MVS.VSE.SNA**	

For TCP/IP, the sender channel name **G** and the receiver channel name **I**, in the preceding table, can be `VSE.sys.tcp` and `sys.VSE.TCP` respectively.

In both cases `sys` represents the remote system name, for example, `OS2`. Therefore, in this case, **G** becomes `VSE.OS2.TCP` and **I** becomes `OS2.VSE.TCP`.

MQSeries for VSE/ESA sender-channel definitions

```
     Local Queue
            Object Type :   L
            Object Name :   OS2                F
              Usage Mode:   T (Transmission)

     Remote Queue
            Object Type :   R
            Object Name :   OS2.REMOTEQ        D
      Remote QUEUE Name :   OS2.LOCALQ         E
      Remote QM    Name :   OS2                C
      Transmission Name :   OS2                F

     Sender Channel
           Channel name :   VSE.OS2.SNA        G
           Channel type :   S (Sender)
Transmission queue name :   OS2                F
                Partner :   OS2                6
                TP Name :   MQTP
```

MQSeries for VSE/ESA receiver-channel definitions

```
     Local Queue
            Object type :   QLOCAL
            Object Name :   VSE.LOCALQ         B
             Usage Mode :   N (Normal)

     Receiver Channel
           Channel name :   OS2.VSE.SNA        I
           Channel type :   R (Receiver)
```

Defining a local queue

1. Run the MQSeries master terminal transaction MQMT.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:51:52           *** Master Terminal Main Menu ***                CIC1
MQWMTP                                                              A004
                         SYSTEM IS ACTIVE

                           1. Configuration

                           2. Operations

                           3. Monitoring

                           4. Browse Queue Records


                        Option: 1




Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Clear/PF3=Exit                          Enter=Select
```

2. Select option 1 to configure.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:52:42           *** Configuration Main Menu ***                  CIC1
MQWMCFG                                                             A004
                          SYSTEM IS ACTIVE

                         Maintenance Options :
                             1. Global System Definition
                             2. Queue Definitions
                             3. Channel Definitions
                             4. Code Page Definitions

                         Display Options     :
                             5. Global System Definition
                             6. Queue Definitions
                             7. Channel Definitions
                             8. Code Page Definitions

                       Option: 2



Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Process  PF2=Return  PF3=Exit
```

3. Select option 2 to work with queue definitions.

```
01/03/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
14:53:16                Queue Main Options                          CIC1
MQWMQUE                                                             A004
                            SYSTEM IS ACTIVE

       Default Q Manager. : VSE.TS.QM1

       Object Type. . . . : L      L = Local Queue
                                   R = Remote Queue
                                   AQ = Alias Queue
                                   AM = Alias Queue Manager
                                   AR = Alias Reply Queue

       Object Name. . . . : VSE.LOCALQ







PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF12=Delete
```

4. Select an Object type of L and specify the name of the queue.
5. Press PF5.

```
01/03/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
14:54:57                Queue Definition  Record                    CIC1
MQWMQUE          QM - VSE.TS.QM1                                    A004

                       Local Queue Definition

Object Name. . . . . . . . : VSE.LOCALQ
Description line 1 . . . . : Test Q
Description line 2 . . . . :

Put Enabled  . . . . . . . : Y   Y=Yes, N=No
Get Enabled  . . . . . . . : Y   Y=Yes, N=No

Default Inbound status . . : A   A=Active,I=Inactive
        Outbound status. . : A   A=Active,I=Inactive

Dual Update Queue. . . . . :

Automatic Reorganize (Y/N) : N   Start Time. : 0000    Interval. . :  0000
VSAM Catalog . . . . . . . :

Record being added   - Press ADD key again.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF10=Queue  PF12=Delete
```

6. Press PF5 again.

```
11/11/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
13:00:08                Queue Extended Definition                   CIC1
MQMMQUE                                                             A005

Object Name: VSE.LOCALQ

General              Maximums                  Events
Type   . . : Local   Max. Q depth . : 00001000  Service int. event: N
File name  : MQFO001  Max. msg length: 00004000  Service interval  : 00000000
Usage  . . : N       Max. Q users . : 00000100  Max. depth event  : N
Shareable  : Y       Max. gbl locks : 00000200  High depth event  : N
                     Max. lcl locks : 00000200  High depth limit  : 000
                                                Low depth event . : N
Triggering                                      Low depth limit . : 000
Enabled  . : N       Transaction id.:
Type . . . :         Program id . . :
Max. starts: 0001    Terminal id  . :
Restart  . : N       Channel name . :
User data  :
           :

Requested record displayed.
PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
            PF9=List  PF10=Queue
```

7. Specify the name of a CICS file to store messages for this queue.
8. If you are creating a transmission queue, specify a **Usage Mode** of `T`, a **Program ID** of `MQPSEND`, and a **Channel Name**< G >.

   For a normal queue specify a **Usage Mode** of `N`.
9. Press PF5 again.

Defining a remote queue

1. Run the MQSeries master terminal transaction MQMT.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:51:52           *** Master Terminal Main Menu ***                CIC1
MQWMTP                                                              A004
                        SYSTEM IS ACTIVE

                          1. Configuration

                          2. Operations

                          3. Monitoring

                          4. Browse Queue Records


                       Option: 1




Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Clear/PF3=Exit                          Enter=Select
```

2. Select option 1 to configure.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:52:42           *** Configuration Main Menu ***                  CIC1
MQWMCFG                                                             A004
                          SYSTEM IS ACTIVE

                         Maintenance Options :
                             1. Global System Definition
                             2. Queue Definitions
                             3. Channel Definitions
                             4. Code Page Definitions

                         Display Options     :
                             5. Global System Definition
                             6. Queue Definitions
                             7. Channel Definitions
                             8. Code Page Definitions

                      Option: 2



Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Process  PF2=Return  PF3=Exit
```

3. Select option 2 to work with queue definitions.

```
01/03/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
15:02:59                Queue Main Options                          CIC1
MQWMQUE                                                             A004
                            SYSTEM IS ACTIVE

       Default Q Manager. : VSE.TS.QM1

       Object Type. . . . : R      L = Local Queue
                                   R = Remote Queue
                                   AQ = Alias Queue
                                   AM = Alias Queue Manager
                                   AR = Alias Reply Queue

       Object Name. . . . : OS2.REMOTEQ









PF2=Return  PF3=Quit  PF4/Enter=Read  PF5=Add  PF6=Update
                                      PF9=List PF12=Delete
```

4. Select an **Object type** of **R** and specify the name of the queue.
5. Press PF5.

```
01/03/2002          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQ212
15:04:25               Queue Definition  Record                     CIC1
MQWMQUE          QM - VSE.TS.QM1                                    A004

                       Remote Queue Definition

Object Name. . . . . . . . : OS2.REMOTEQ
Description line 1 . . . . : Test remote queue on OS/2
Description line 2 . . . . :

Put Enabled  . . . . . . . : Y   Y=Yes, N=No
Get Enabled  . . . . . . . : Y   Y=Yes, N=No

Remote Queue Name. . . . . : OS2.LOCALQ
Remote Queue Manager Name. : OS2
Transmission Queue Name. . : OS2




Record being added   - Press ADD key again.
PF2=Return  PF3=Quit  PF4/Enter=Read   PF5=Add  PF6=Update
                                       PF9=List PF12=Delete
```

6. Specify a remote queue name, remote queue manager name, and transmission queue name.
7. Press PF5.

Defining a SNA LU 6.2 sender channel

1. Run the MQSeries master terminal transaction MQMT.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:51:52           *** Master Terminal Main Menu ***                CIC1
MQWMTP                                                              A004
                        SYSTEM IS ACTIVE

                          1. Configuration

                          2. Operations

                          3. Monitoring

                          4. Browse Queue Records


                       Option: 1




Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Clear/PF3=Exit                          Enter=Select
```

2. Select option 1 to configure.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:52:42           *** Configuration Main Menu ***                  CIC1
MQWMCFG                                                             A004
                          SYSTEM IS ACTIVE

                         Maintenance Options :
                             1. Global System Definition
                             2. Queue Definitions
                             3. Channel Definitions
                             4. Code Page Definitions

                         Display Options     :
                             5. Global System Definition
                             6. Queue Definitions
                             7. Channel Definitions
                             8. Code Page Definitions

                      Option: 3


Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Process  PF2=Return  PF3=Exit
```

3. Select option 3 to work with channel definitions.

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2          TSMQBD
15:19:47                    Channel Record           DISPLAY        CIC1
MQMMCHN                                                             A000
Channel  : VSE1.SNA.AIX1
 Desc. . :
 Protocol: L (L/T)     Type : S (Sender/Receiver/svrConn)  Enabled : Y

Sender
 Remote TCP/IP port . . . . : 00000      LU62 Allocation retry num : 00000003
 Get retry number . . . . . : 00000003   LU62 delay fast (secs). . : 00000005
 Get retry delay (secs) . . : 00000015   LU62 delay slow (secs). . : 00000001
 Convert msgs(Y/N). . . . . : N
 Transmission queue name. . : VSE1.AIX1.XQ1
 TP name. . :

Sender/Receiver
 Connection : AIX1
 Max Messages per Batch . . : 000050     Message Sequence Wrap . . : 999999
 Max Message Size   . . . . : 0005000    Dead letter store(Y/N)  . : Y
 Max Transmission Size  . . : 032000     Split Msg(Y/N)  . . . . . : N
 Max TCP/IP Wait  . . . . . : 000300

Channel record displayed.
F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

4. Complete the parameter fields as indicated, specifically the fields **Channel name**< **G** >, **Channel type**, **Connection ID**, **Remote task ID**, and **Transmit queue name**< **F** >.

   All other parameters can be entered as shown.

   Note that the default value for **sequence number wrap** is 999999, whereas for Version 2 MQSeries products, this value defaults to 999999999.

5. Press PF5.

Defining a SNA LU 6.2 receiver channel

1. Run the MQSeries master terminal transaction MQMT.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:51:52           *** Master Terminal Main Menu ***                CIC1
MQWMTP                                                              A004
                        SYSTEM IS ACTIVE

                          1. Configuration

                          2. Operations

                          3. Monitoring

                          4. Browse Queue Records


                       Option: 1




Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Clear/PF3=Exit                          Enter=Select
```

2. Select option 1 to configure.

```
01/03/2002        IBM MQSeries for VSE/ESA Version 2.1.2            TSMQ212
14:52:42           *** Configuration Main Menu ***                  CIC1
MQWMCFG                                                             A004
                          SYSTEM IS ACTIVE

                         Maintenance Options :
                             1. Global System Definition
                             2. Queue Definitions
                             3. Channel Definitions
                             4. Code Page Definitions

                         Display Options     :
                             5. Global System Definition
                             6. Queue Definitions
                             7. Channel Definitions
                             8. Code Page Definitions

                      Option: 3


Please enter one of the options listed.
    5686-A06 (C) Copyright IBM Corp. 1998, 2002. All Rights Reserved.
Enter=Process  PF2=Return  PF3=Exit
```

3. Select option 3 to work with channel definitions.

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
15:19:47                    Channel Record           DISPLAY         CIC1
MQMMCHN                                                              A000
Channel  : AIX1.SNA.VSE1
 Desc. . :
 Protocol: L (L/T)     Type : R (Sender/Receiver/svrConn)  Enabled : Y

Sender
 Remote TCP/IP port . . . . : 00000      LU62 Allocation retry num : 00000003
 Get retry number . . . . . : 00000000   LU62 delay fast (secs). . : 00000005
 Get retry delay (secs) . . : 00000000   LU62 delay slow (secs). . : 00000001
 Convert msgs(Y/N). . . . . : N
 Transmission queue name. . :
 TP name. . :

Sender/Receiver
 Connection :
 Max Messages per Batch . . : 000050     Message Sequence Wrap . . : 999999
 Max Message Size   . . . . : 0005000    Dead letter store(Y/N)  . : Y
 Max Transmission Size  . . : 032000     Split Msg(Y/N)  . . . . . : N
 Max TCP/IP Wait  . . . . . : 000300

Channel record displayed.
F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

4. Complete the parameter fields as indicated, specifically the field **Channel name**< **L** >.

   All other parameters can be entered as shown.
5. Press PF5.

Defining a TCP/IP sender channel

To define a TCP/IP sender channel, carry out the following procedure:

1. Run the MQSeries master terminal transaction MQMT.
2. Select option 1 to configure.
3. Select option 3 to work with channel definitions. The screen shown in Figure 64 is displayed:

```
11/05/2003          IBM MQSeries for VSE/ESA Version 2.1.2           TSMQBD
 15:19:47                    Channel Record           DISPLAY         CIC1
 MQMMCHN                                                              A000
 Channel  : VSE1.TCP.NT1
  Desc. . :
  Protocol: T (L/T)     Type : S (Sender/Receiver/svrConn)  Enabled : Y

 Sender
  Remote TCP/IP port . . . . : 01414      LU62 Allocation retry num : 00000000
  Get retry number . . . . . : 00000003   LU62 delay fast (secs). . : 00000000
  Get retry delay (secs) . . : 00000030   LU62 delay slow (secs). . : 00000000
  Convert msgs(Y/N). . . . . : N
  Transmission queue name. . : VSE1.NT1.XQ1
  TP name. . :

 Sender/Receiver
  Connection : NT1SERV
  Max Messages per Batch . . : 000050     Message Sequence Wrap . . : 999999
  Max Message Size   . . . . : 0005000    Dead letter store(Y/N)  . : Y
  Max Transmission Size  . . : 032766     Split Msg(Y/N)  . . . . . : N
  Max TCP/IP Wait  . . . . . : 000300

 Channel record displayed.
 F2=Return PF3=Quit PF4=Read PF5=Add PF6=Upd PF9=List PF10=SSL PF11=Ext PF12=Del
```

Figure 64. Channel configuration panel

4. Complete the parameter fields as follows:
   - Channel name – **G** on the configuration worksheet.
   - Partner – should contain either the domain name or the IP address of the remote host, for example NTSERV1 or 1.20.33.44.
   - Port – the port number must match the port number configured for the remote host. This is configured in the global system definition of the remote host. The default port number for MQSeries for VSE/ESA is 1414.
   - Transmission queue name – **F** on the configuration worksheet.
   - Protocol – enter `T` for TCP/IP
   - Channel type – enter `S` for sender

   Notes:

   1. The TP Name is not used by TCP/IP channels.
   2. Ensure that the parameter field values match the values of the receiver channel definition of the same name on the remote host.
5. Press PF5 (Add) to add the new channel definition.

Defining a TCP/IP receiver channel

To define a TCP/IP receiver channel, carry out the following procedure:

1. Run the MQSeries master terminal transaction MQMT.
2. Select option 1 to configure.
3. Select option 3 to work with channel definitions. The screen shown in Figure 64 is displayed.
4. Complete the parameter fields as follows:
   - Channel name – **G** on the configuration worksheet.
   - Protocol – enter `T` for TCP/IP
   - Channel type – enter `R` for receiver.

Notes:

1. The Partner and Port are not required for a TCP/IP receiver channel.
2. The TP Name is not used by TCP/IP channels.
3. Ensure that the parameter field values match the values of the sender channel definition of the same name on the remote host.

5. Press PF5 (Add) to add the new channel definition.

Appendix F. MQSeries clients

An MQSeries client is an MQSeries system that does not include a queue manager. The MQSeries client code directs MQI calls from applications running on the client system to a queue manager on an MQSeries server system to which it is connected.

This appendix provides information about MQSeries clients that is specific to MQSeries for VSE/ESA V2.1. It should be used in conjunction with the *MQSeries Clients* book.

Client support

MQSeries for VSE/ESA V2.1 can function as an MQSeries server system to all MQSeries clients that can connect to the server using the TCP/IP protocol. However, there is no MQSeries for VSE/ESA V2.1 client.

All current clients are available from the MQSeries Web site at:

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

When an MQSeries client connects to a queue manager on MQSeries for VSE/ESA V2.1, the client can issue MQI calls as if the queue manager were local to the client program. Valid MQI calls are:

MQCONN	Connect queue manager
MQOPEN	Open message queue
MQGET	Get message
MQPUT	Put message
MQPUT1	Put one message
MQINQ	Inquire about object attributes
MQCLOSE	Close object
MQDISC	Disconnect queue manager

Security considerations

If MQSeries for VSE/ESA Version 2.1.2 is installed without the security feature, client connections are not authenticated. Channel security exits are not supported by Version 2.1.2.

If MQSeries for VSE/ESA Version 2.1.2 is installed with the security feature, MQSeries authenticates client connections via the MQ_USER_ID and MQ_PASSWORD environment variables defined under the client environment session. This means that MQSeries for VSE/ESA only provides security for clients that support these environment variables.

Once a client connection is established, all MQI calls are verified as if they were executed by the user identified by the MQ_USER_ID variable. See Chapter 11, “Security,” on page 285 for more information.

Client code-page conversion tables

For client/server code-page conversion, the MQSeries server uses the Language Environment (LE) code set conversion facilities:

© Copyright IBM Corp. 1995, 2004

genxlt utility
Generates a translation table for use by the iconv utility and the iconv functions to perform code page conversion.

iconv utility
Converts a file from one code set encoding to another.

iconv functions
Functions used by the MQSeries server to perform all code-page conversions.

The `genxlt` utility generates object output that is link-edited to produce a phase. Each phase represents a translation table between two code pages. Consequently, the range of supported client/server code pages is determined by the number of phases generated using the `genxlt` utility.

The `genxlt` and `iconv` language environment utilities are fully documented in the *C Run-Time Programming Guide*, which documents all the supplied code-set converters.

You are **strongly recommended** to read the section on code-page conversion in the *C Run-Time Programming Guide*. If you need code conversion for pages that are not provided, you can edit the appropriate source-code modules and build the converters.

Appendix G. System messages

This appendix describes the messages issued by MQSeries.

MQSeries generates both internal and external messages. Internal messages are generated when an application program activates MQSeries and an abnormal condition occurs.

These messages are stored on the system log queue when it is available; otherwise, the CICS CSMT Transient Data (TD) queue is used.

API system messages

These messages consist of five lines of text, each with a maximum of 78 characters, together with two lines of error code information as follows:

Line 1 –

```
MQInnnnnns PRG:pppppppp TRN:tttt TRM:rrrr TSK:ccccc mm/dd/yy hh:mm:ss
```

Where:

`nnnnnn` MQSeries message code – see "MQSeries message codes" on page 468

`s` Severity

Severity values have the following meanings:

- I An information message. No error has occurred.
- W A warning message. A condition has been detected of which you should be aware. You may need to take further action.
- E An error message. An error has been detected that the system typically corrects. However, you may have to intervene.
- C A critical error message. An error has been detected that may severely affect user or system operation. This requires your immediate intervention.

`pppppppp`
CICS Program name

`tttt` CICS Transaction code

`rrrr` CICS Term ID

`ccccc` CICS Task ID

`mm/dd/yy`
Date

`hh:mm:ss`
Time

Line 2 – Textual description of message

Line 3 – Queue name, if available

Line 4 – Channel name, if available

Line 5 – Detail of message (optional)

Line 6 –

```
EIBFN:fff EIBRCODE:rrrrrrrrrrrr EXEC LINE: lllll
```

© Copyright IBM Corp. 1995, 2004

Where:
fff EIBFN value at time of condition
rrrrrrrr
EIBRCODE
lllll The DEBUG CICS command number

Line 7 –

```
EIBRESP: rrrrrrrr EIBRESP2: ssssssss EIBRSRCE:cccccccc ABCODE: aaaa
```

Where:
rrrrrrrr
EIBRESP
ssssssss
EIBRESP2
cccccccc
EIBRSRCE
aaaa CICS ABENDCODE

MQSeries message definitions

Each MQSeries message provides the following information:

Explanation: This section explains what the message or code means, why it occurred, and what caused it.

Function This section indicates which modules issued the message, to assist in diagnosing problems.

Operator action If an operator response is necessary, this section describes what the appropriate responses are, and what their effect is. If this information is omitted, no operator response is required.

System action This part describes what is happening as a result of the condition causing the message or code. If this information is omitted, no system action is taken.

MQSeries messages

MQSeries system messages are numbered 000000 through 900000, and they are listed in this book in numeric order. However, not all numbers have been used, therefore, the list is not continuous.

MQSeries console messages are numbered from MQI0001 onwards, and they are listed in this book in numeric order; see “Console Messages” on page 496.

MQSeries message codes

000000I Queue manager started

Explanation: The local queue manager has been started.

User Response: None.

System Action: The queue manager is available for queuing services.

000001I Queue manager stop requested

Explanation: A request to stop the queue manager has been issued.

User Response: None.

System Action: Queue manager services are stopping.

000002I Queue manager stopped

Explanation: The local queue manager has been stopped.

User Response: None.

System Action: Queue manager services are unavailable until the system is restarted.

000003E Channel Message Sequence Number error

Explanation: The received MSN does not match the expected MSN.

User Response:

1. Review the LOCAL MSN and the REMOTE MSN in the detail portion of the message.
2. Identify the cause (proper running should preclude this occurance).
3. Reset the appropriate MSN so that the sender and receiver channel MSN's are equal.
4. Restart communication.

System Action: Fatal error - Communication is terminated.

000004W Synch MSG duplicate

Explanation: The received message may be duplicated.

User Response: None.

System Action: Continue on negotiating.

000007I LU62 session started

Explanation: A communication session was started by MQPRECV.

User Response: None.

System Action: None.

000009E Channel wrap value negotiation mismatch

Explanation: A sent WRAP value did not match that of the of the receiver.

User Response: Review the SENT and RECEIVED match values and change one side to match the other.

System Action: Channel communication ends.

000010E LU62 FREE error

Explanation:

For Program MQPRECV
Upon completion of a RECEIVE command the EIBFREE and the EIBERR fields are both not equal to low values.

For Program MQPSEND
As a Server upon completion of a RECEIVE command at least one of EIBERR, EIBRECV and EIBFREE does not equal to low values.

As a Server or Sender upon receipt of an acknowledgement of messages sent the EIBFREE is not equal to low values and the EIBERR is equal to low values.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000011E LU62 EIB error

Explanation: For Program MQPSEND

1. As a Server upon completion of a RECEIVE the EIBERR not equal to low values.
2. As a Server or Sender upon receipt of an acknowledgement of messages sent, the EIBERR is not equal to low values.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000012E LU62 STAT error For Program MQPSEND -

Explanation: As a Server or Sender upon receipt of an acknowledgement of messages sent, the EIBRECV is not equal to low values.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000013E LU62 ALLOC error

Explanation: For Program MQPSEND

- As a Sender upon completion of an ALLOCATE command, EIBRCODE is not equal to low values and all retries have been performed.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000014W LU62 ALLOC RETRY error For Program MQPSEND -

Explanation: As a Sender upon completion of an ALLOCATE command, EIBRCODE is not equal to low values and all retry attempts have not been performed.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.

System Action: Non-Fatal error - Allocation is retried until allocation is successful or the retry count equals zero.

000015E LU62 CONN error

Explanation: For Program MQPSEND

- As a Sender upon completion of a CONNECT PROCESS command, EIBRCODE is not equal to low values.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000016E LU62 SEND error

Explanation: For Program MQPSEND

- As a Sender or Server upon completion of a SEND command, EIBRCODE is not equal to low values.

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000017I Remote deallocation or communication shutdown

Explanation: For Program MQPSEND or MQPRECV

- The remote MCA deallocated the conversation or closed an active IP socket. Alternatively, a request to shutdown the MQ communications subsystem was issued (via MQ shutdown).

User Response: Normally, this is an informational message and no additional user action is required.

However, in the event that the deallocation has occurred due to a remote system failure, intervention may be required on the remote system.

System Action: Communication is terminated.

000023E Invalid error data

Explanation: For Program MQPSEND or MQPRECV

- The Sender or Receiver MCA received an error data transmission that contained an unrecognized error code.

User Response: Review System Log or error TD Queue for messages prior to this message. Proper running should preclude this occurrence.

System Action: Fatal error - Communication is terminated.

000024E Invalid MSN value received during negotiation

Explanation: (Reserved)

000025E Fatal response type

Explanation: (Reserved)

000026E Recoverable response type

Explanation: (Reserved)

000029E Parser MSN error

Explanation: (Reserved)

000030E Parser type error

Explanation: (Reserved)

000031E Parser PDM error

Explanation: (Reserved)

000032E Parser SID error

Explanation: (Reserved)

000033E Parser PN error

Explanation: (Reserved)

000034E Parser KEY error

Explanation: (Reserved)

000035E Parser APID error

Explanation: (Reserved)

000038E Parser ORG DT error

Explanation: (Reserved)

000039E Parser ORIG MSN error

Explanation: (Reserved)

000040E Parser BODY error

Explanation: (Reserved)

000041E Parser status error

Explanation: The received message does not have the proper status value.

User Response: Review System Log or error TD Queue for messages prior to this message. Proper running should preclude this occurrence. Investigate sender process for programming error.

System Action: Fatal error - Communication is terminated.

000042E Parser length error

Explanation: The received message does not have the proper length value.

User Response: Review System Log or error TD Queue for messages prior to this message. Proper running should preclude this occurrence. Investigate sender process for programming error.

System Action: Fatal error - Communication is terminated.

000051E Queue connection error

Explanation: The QM cannot be connected to.

User Response: Review System Log for associated error messages. Ensure that the queue manager has not been shutdown during communication activity.

System Action: Fatal error - Communication is terminated.

000052E Queue open error

Explanation: The Sender or Receiver MCA could not open a source or target queue.

User Response:

1. For a sender channel, check that the associated transmission queue is valid and enabled. For a

receiver channel, check that the intended target queue and the system dead letter queue are valid and enabled.

2. Review the fields in this error message:
   QUEUE ID
   Transmission queue name that failed.
   CHANNEL ID
   Channel name that was connected. This channel identifies the corresponding transmission queue.
   Last line of error message
   Reason code returned from queuer and corresponding description.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000053E Queue GET error

Explanation: The Server or Sender could not get a message from the associated transmission queue even if there is (are) message(s) in the transmission queue.

User Response:

1. Review the following fields in the error message:
   QUEUE ID
   Transmission queue name that failed.
   CHANNEL ID
   Channel name that was connected. This channel identifies the corresponding transmission queue.
   Last line of error message
   Reason code returned from queuer and corresponding description.
2. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000054E Queue PUT error

Explanation: The RECEIVER could not put a message to an application queue.

User Response:

1. Review the following fields in the error message:
   QUEUE ID
   Application queue name that failed.
   CHANNEL ID
   Channel name that was connected.
   Last line of error message
   Reason code returned from queuer and corresponding description.
2. User action is based upon returned reason code:
   Reason code equals MQRC-Q-FULL (2053) or MQRC-Q-SPACE-NOT-AVAILABLE (2056)
   Destination application queue was full and the message was placed on the dead letter queue. Determine if destination queue should be expanded to accommodate more messages or an alternate destination used.
   All other Reason codes
   Correct problem and restart communication.

System Action: There are two possible system actions based upon reason code returned from queuer:
Reason code equals MQRC-Q-FULL or MQRC-Q-SPACE-NOT-AVAILABLE
Non-Fatal error - communication will proceed normally after first putting failed put message on dead letter queue.
All other Reason codes
Fatal error - Communication is terminated.

000055E Queue PUT1 error

Explanation: The RECEIVER could not put a message to the dead letter queue.

User Response:

1. Review the following fields in the error message:
   QUEUE ID
   The dead letter queue name that failed.
   CHANNEL ID
   Channel name that was connected.
   Last line of error message
   Reason code returned from queuer and corresponding description.
2. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

000056W Queue CLOSE error

Explanation: The RECEIVER could not close an application queue.

User Response:

1. Review the following fields in the error message:
   QUEUE ID
   Application queue name that failed.
   CHANNEL ID
   Channel name that was connected.
   Last line of error message
   Reason code returned from queuer and corresponding description.
2. Investigate problem

System Action: Non-Fatal error - communication will proceed normally. (The un-closed resources, however, will result in a "garbage collection" mechanism be triggered at a proper time to close the un-closed resources).

000057E Queue DISC error

Explanation: An error has occurred to DISCONNECT the connecting Queue Manager.

User Response: Review System Log or error TD Queue for messages prior to this message. Proper running should preclude this occurrence. Investigate sender process for program error.

System Action: Fatal error - Communication is terminated.

000060E Unexpected queue error

Explanation: An unexpected error occurred during queue processing.

User Response: This can occur if the queue manager is shutdown while queue manager connections are active.

System Action: Fatal error - queue processing is terminated.

000080E Receiver return LON status

Explanation: (Reserved)

000081E Receiver return LON type

Explanation: (Reserved)

000091E SIDRC return format

Explanation: (Reserved)

000100I Function started

Explanation: The requested function has been started

User Response: None.

System Action: Function is started

000110I Queue modification requested

Explanation: A request was issued to modify a queue.

User Response: None.

System Action: If the request is successful, the queue is modified.

000112I Queue message modification requested

Explanation: A request was issued to modify messages on a queue.

User Response: None.

System Action: If the request is successful, the queue messages are modified.

000114I Channel modification requested

Explanation: A request was issued to modify a channel.

User Response: None.

System Action: If the request is successful, the channel is modified.

000116I Queue manager modification requested

Explanation: A request was issued to modify the queue manager.

User Response: None.

System Action: If the request is successful, the queue manager is modified.

001000I Function completed

Explanation: The requested function has been completed

User Response: None.

System Action: Function is completed.

001090E Function not completed

Explanation: The requested function was terminated because of error. The function was not completed.

User Response: Review the associated message prior to this one.

System Action: Function is terminated with error.

005000I Channel connected

Explanation: Channel connection is successful.

User Response: None.

System Action: platform negotiation will begin.

002000I System monitor has detected an unowned connection

Explanation: The queue manager housekeeping task has identified a MQI connection for a task that is no longer active.

User Response: None.

System Action: The connection is ended, and associated resources are freed.

002010I System monitor has detected an inactive object handle

Explanation: The queue manager housekeeping task has identified an object handle (HOBJ) for an MQI connection that no longer exists.

User Response: None.

System Action: The object handle is closed, and associated resources are freed.

005001I Channel negotiations accepted

Explanation: Channel has completed negotiation with the other platform.

User Response: None.

System Action: Message queue will be opened.

005002I Channel queue opened

Explanation: Channel queue has been opened successfully.

User Response: None.

System Action: Message transfer will begin.

005003I LU 6.2 connection established

Explanation: LU 6.2 connection established.

User Response: None.

System Action: LU 6.2 conversation will begin.

005004I Channel receiver allocated

Explanation: (Reserved)

005005I Channel queue empty

Explanation: Sender finds the queue is empty

User Response: None.

System Action: Transmission queue will be closed and the Channel will be disconnected and shutdown after number of get retries exhausted.

005006I Channel queue closed

Explanation: Channel has successfully closed queue.

User Response: None.

System Action: Channel will be disconnected.

005007I Channel disconnected

Explanation: Channel has been disconnected from the other platform.

User Response: None.

System Action: Channel will be shutdown.

005008I Channel shutdown

Explanation: Channel has been completely shutdown.

User Response: None.

System Action: Channel is marked INACTIVE.

005009I Channel shutdown request sent

Explanation: (Reserved)

006003I TCP/IP channel connected

Explanation: TCP/IP connection established.

User Response: None.

System Action: TCP/IP conversation will begin.

006007I TCP/IP session started

Explanation: A communication session was started by the MQI Receiver MCA.

User Response: None.

System Action: TCP/IP conversation will begin.

006010E TCP/IP connection broken

Explanation: A TCP/IP connection terminated prematurely, possibly due to incomplete data from remote partner, or the premature closure of an active TCP/IP socket.

User Response:

1. Review System Log or error TD Queue for messages prior to this message.
2. Check host TCP/IP services are active.
3. Check TCP/IP error log on remote system.
4. Correct problem and restart channel.

System Action: Fatal error - Communication is terminated.

006013E TCP/IP storage allocation error For Program MQPTCPSV -

Explanation: The MQ/Server program was unable to allocate memory.

User Response: This error occurs when there is insufficient memory resources available. Check that other processes are not erroneously allocating memory, or rerun the client/server conversation when the VSE host is less busy.

System Action: Fatal error - Communication is terminated.

006015E TCP/IP connection error For Program MQPSEND -

Explanation: As a Sender upon completion of a CONNECT request, received an invalid return code.

User Response:

1. Review System Log or error TD Queue for messages prior to this message.
2. Check that the sender channel port number matches the listener port of the receiver.
3. Check that the TCP Partner is a valid host or IP address n.n.n.n of the receiving system.
4. Correct problem and restart communication.

System Action: Fatal error - Communication is not established.

006016E TCP/IP send error

Explanation: As a Sender or Server upon completion of a data send request received an invalid return code.

User Response:

1. Review System Log or error TD Queue for messages prior to this message.
2. Produce an auxiliary trace to determine the return code returned from the send.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

006017E TCP/IP receive/respond error

Explanation: An attempt to receive TCP/IP data failed or the response from a remote system was other than as expected.

User Response:

1. Review System Log or error TD Queue for messages prior to this message.
2. Check the TCP/IP error log on the remote system to see if the channel was closed prematurely.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

006020E TCP/IP socket error

Explanation: An error occurred when opening or setting the attributes of a TCP/IP socket.

User Response:

1. Check that TCP/IP is installed and running.
2. Check that the TCP/IP phase is cataloged in a library that is concatenated ahead of SCEEBASE in the LIBDEF of the CICS startup deck.

System Action: Fatal error - communication cannot start.

006021E TCP/IP transport error

Explanation: A TCP/IP conversation ended due to a transport protocol error, or the use of a conversation was attempted before it was established.

User Response:

1. Check that TCP/IP is running.
2. Attempt to restart the conversation.

System Action: Fatal error - communication is terminated.

006022C TCP/IP listener bind error

Explanation: The MQ Listener program attempted to bind a port number to a TCP/IP socket and was unsuccessful.

User Response:

1. Check that the Listener is not already running.
2. Check that another application is not using the port number configured for the Listener.

System Action: Fatal error - Listener cannot start.

006023C TCP/IP listener accept error

Explanation: The MQ Listener program attempted to accept a remote conversation and failed.

User Response:

1. This is not an error if the remote program terminated before the conversation was accepted.
2. Check that TCP/IP is running.
3. Restart the Listener.

System Action: Error - Listener is terminated.

006024C TCP/IP listener error

Explanation: The MQ Listener program terminated due to an unexpected error.

User Response:

1. Check that TCP/IP is running.
2. Examine the system log for associated error messages.
3. Restart the Listener.

System Action: Fatal error - Listener is terminated.

006025I TCP/IP listener stopped

Explanation: The MQ Listener program terminated normally or due to an error.

User Response:

1. Check the previous log entries for error messages. If no previous error messages then the Listener terminated normally.
2. Restart the Listener when appropriate.

System Action: Listener program terminates.

006026E TCP/IP invalid channel type

Explanation: The channel type value in a channel definition is invalid.

User Response:

1. Check channel definition documentation for valid channel type values. Then update the channel definition appropriately.

System Action: Error - Channel cannot be started.

006027E TCP/IP channel negotiation failed

Explanation: The conversation initial negotiation failed.

User Response:

1. Check that the local and remote channel definitions are compatible.

System Action: Error - channel cannot be started.

006028E TCP/IP channel protocol error

Explanation: The protocol type value in a channel definition is invalid.

User Response:

1. Check channel definition documentation for valid protocol type values. Then update the channel definition appropriately.

System Action: Error - Channel cannot be started.

006029E TCP/IP connect failed

Explanation: An attempt to establish a TCP/IP connection failed.

User Response:

1. Check channel definition to ensure the hostname or IP address and port number are valid for the intended remote host.
2. Check that the remote system's listener process if active.

System Action: Fatal error - conversation cannot be started.

006030E TCP/IP unknown remote channel name

Explanation: The channel name used in a remote conversation does not exist on the remote host.

User Response:

1. Create the channel definition on the remote host.

System Action: Error - communication is terminated.

006031E TCP/IP remote queue manager not available

Explanation: The queue manager identified in for a remote host is currently unavailable.

User Response:

1. Start the remote queue manager on the remote host. :
2. Retry the channel.

System Action: Error - communication is terminated.

006032W TCP/IP channel stopped by operator/application

Explanation: The channel being used in a remote conversation has been disabled by an operator or application.

User Response:

1. Restart the channel when appropriate.

System Action: Error - communication is terminated.

006033E TCP/IP channel not active

Explanation: An attempt was made to use a channel that is not currently started.

User Response:

1. Start the channel when appropriate.

System Action: Error - communication is terminated.

006034I TCP/IP channel stopped

Explanation: The channel stopped normally or due to an error.

User Response:

1. Check previous log messages for errors. If there are no errors, then the channel stopped normally.
2. Restart the channel when appropriate.

System Action: Channel activity is terminated.

006035C TCP/IP syncpoint failed

Explanation: An attempt to perform a CICS SYNCPOINT failed during a TCP/IP conversation.

User Response:

1. Check the system log for previous error messages. :
2. Check CICS logs for possible errors or insufficient resources.

System Action: Transaction changes are rolled back to the beginning of the current unit of work.

006036W TCP/IP message put to DLQ

Explanation: A message could not be delivered and was instead written to the system dead letter queue.

User Response:

1. Examine the dead letter queue for the undelivered message.

System Action: Message written to system dead letter queue.

006037E TCP/IP invalid remote channel type

Explanation: The channel type value in a channel definition on a remote host is invalid.

User Response:

1. Check channel definition documentation for valid channel type values. Then update the channel definition on the remote host appropriately.

System Action: Error - Channel cannot be started.

006038W TCP/IP transmission queue get inhibited

Explanation: The transmission queue identified by a channel definition has GET INHIBIT enabled.

User Response:

1. Disable GET INHIBIT on the transmission queue.

System Action: Error - messages cannot be retrieved and sent.

006039E TCP/IP remote channel unavailable

Explanation: The channel identified for a remote conversation is unavailable.

User Response:

1. Start the channel on the remote host.

System Action: Error - channel cannot be started.

006040E TCP/IP channel abnormally ended

Explanation: The channel involved in a current conversation was terminated with an error, possible due a remote partner ending the conversation prematurely.

User Response:

1. Check the system log for previous error messages.

System Action: Fatal error - communication is terminated.

006041I TCP/IP listener started

Explanation: The MQ Listener program started.

User Response: None.

System Action: MQ Listener ready to accept remote connections.

006042I TCP/IP server started

Explanation: An MQ Server program instance started in response to a remote client connection.

User Response: None.

System Action: MQ Server ready to process client requests.

006043I TCP/IP server stopped

Explanation: An MQ Server program instance stopped normally or due to an error.

User Response:

1. Check previous system log messages to determine whether this is a normal termination or due to an error.

System Action: Client conversation terminated.

006044E TCP/IP bad server commarea

Explanation: An MQ Server program instance was started with an invalid commarea.

User Response:

1. Check the MQ Server program is not being started by an application other than the MQ Receiver MCA.

System Action: Fatal error - MQ Server program terminated.

006045W TCP/IP server error

Explanation: An error occurred during an MQ client conversation.

User Response:

1. Check the system log for previous error messages and respond to these.

System Action: Client conversation completed with error(s).

006047W Data conversion code page error

Explanation: Data conversion from source code page to target code page is not supported.

User Response: Check the requested code pages are compatible and change if necessary.

System Action: The message data is not converted.

006050W Data conversion source code page error.

Explanation: The value in the source code page is an unknown value.

User Response: Check the source code page is valid.

System Action: The message data is not converted.

006053W Data conversion target code page error

Explanation: The value in the target code page is an unknown value.

User Response: Check the target code page is valid.

System Action: The message data is not converted.

006054W Default data conversion code page error

Explanation: Default data conversion between the specified code pages is not supported.

User Response: Check the default code pages specified in the code page definition panel are compatible.

System Action: The message data is not converted.

006055W Data conversion IMS string error

Explanation: The passed IMS string is the wrong length for data conversion.

User Response: Change the IMS string length.

System Action: The IMS message data is not converted.

006057W Data conversion PCF header bad length

Explanation: The passed PCF header is the wrong length.

User Response: Change the PCF header string length. The buffer is too small for a complete PCF header.

System Action: The PCF message data is not converted.

006058E Data conversion bad ICONV return code for LE conversion

Explanation: Data conversion failed.

User Response:

1. Ensure LE/VSE code page is available.
2. Review LE ICONV family services and ensure product features are installed and functional.

System Action: The message data is not converted.

006059I Data conversion target code page set to source code page

Explanation: The target code was set to null.

User Response: Ensure the target code page is correctly set.

System Action: Data conversion continues.

006060W Data conversion is not supported

Explanation: Data conversion is not supported between the specified code pages.

User Response: Change the code pages to a pair which are supported for LE data conversion.

System Action: Conversion fails.

006063C Data conversion error finding MQ internal control block

Explanation: MQ/VSE is not initialized correctly.

User Response:

1. Stop and restart MQ Series. Check MQ and CICS logs for possible errors such as SOS.
2. If product maintenance has been recently applied, review cover letter for installation steps that may have been missed.

System Action: Data conversion ends.

006100E TCP/IP SSL initialization failed

Explanation: Attempt to establish a secure socket connection using SSL failed.

User Response:

1. Check that remote system requesting the secure socket connection uses standard SSL protocol and valid PKI certificate.
2. For a sending system, check that local SSL services are installed and configured correctly.

System Action: Channel will not initialize.

006101E TCP/IP SSL cipher specification error

Explanation: Secure socket connection abandoned due to channel cipher specification mismatch.

User Response:

1. Check that the remote system requesting the secure socket connection identifies a cipher specification that matches the cipher specification of the local receiver channel.
2. Check that local SSL services support the channel cipher specification.

System Action: Channel is terminated.

006102E TCP/IP SSL peer attribute error

Explanation: Secure socket connection abandoned due to channel peer attribute mismatch with partner certificate details.

User Response:

1. Check that the SSLPEER specification of the local channel identifies the certificate details of the remote system communications partner. (Remember that values are case sensitive).
2. Check that the remote client or sender MCA is an authorized SSL partner.

System Action: Channel is terminated.

006103E TCP/IP SSL client authentication error

Explanation: Secure socket connection abandoned due to remote system failing to provide a valid certificate during channel initialization.

User Response:

1. If the receiver channel specifies SSLCAUTH is REQUIRED, then the remote client or sender MCA must provide a valid certificate during channel/SSL initial negotiation.
2. If the receiver channel does not require a client certificate, then the receiver channel can set the SSLCAUTH to optional.
3. If the receiver channel requires a peer name match, then the remote system must provide a certificate during channel initialization.

System Action: Channel is terminated.

006999E TCP/IP unexpected error

Explanation: An unexpected error has occurred.

User Response:

1. Check the system log for previous error messages.
2. Check the QCODE in this error message. This should be a numeric code that will be meaningful to MQ system support.

System Action: Unknown.

007000I PCF command server started

Explanation: The PCF command server has been started.

User Response: None.

System Action: PCF command server ready to process PCF commands from the system command queue.

007001W PCF command server not started

Explanation: The PCF command server could not be started.

User Response: Check the following:

1. Queue manager is active.
2. Command server not already active.
3. System log for further error messages.

System Action: PCF command server instance cannot start.

007002W PCF command server not started

Explanation: The PCF command server could not be stopped because it is not running.

User Response: None.

System Action: None.

007003I PCF command server terminated by request

Explanation: The PCF command server has terminated due to an operational request.

User Response: Restart the command server when necessary.

System Action: The PCF command server is nolonger available to processes PCF commands from the command queue.

007004I PCF command server termination requested

Explanation: An operational request to terminate the PCF command server has been received.

User Response: None.

System Action: The PCF command server will terminate, and can be restarted when necessary.

007005W PCF command server already running

Explanation: An request to start the PCF command server was received when the server is already running.

User Response: None.

System Action: Only one instance of the command server can be running at any time. The initial instance will continue to process command messages until terminated.

007006C Insufficient storage for PCF command server

Explanation: An attempt to allocate CICS storage for the PCF command server during initialization failed.

User Response: The command server attempts to allocate approximately 2k of CICS storage during initialization. Ensure the relevant CICS region has sufficient storage resources.

System Action: The PCF command server cannot be started.

007007C PCF command server cannot process command queue

Explanation: An attempt by the PCF command server to issue an MQGET from the system command queue failed.

User Response: Check the status of the system command queue to ensure it is enabled and available for processing.

System Action: The PCF command server is terminated.

007008C PCF command server cannot connect to queue manager

Explanation: An attempt by the PCF command server to issue an MQCONN to establish an MQI session with the local queue manager failed.

User Response: Check the status of the queue manager to ensure it is active and available for MQI connectivity.

System Action: The PCF command server is terminated.

007009C PCF command server cannot open command queue

Explanation: An attempt by the PCF command server to issue an MQOPEN of the system command queue failed.

User Response: Check the status of the system command queue to ensure it is enabled and available for processing.

System Action: The PCF command server is terminated.

007010W PCF command server stopped due to system quiescing

Explanation: The PCF command server has detected that the queue manager is quiescing or stopping.

User Response: No action is required if the queue manager has been stopped intentionally. If not, the cause of the queue manager stopping should be investigated, and the PCF command server restarted if a manual start is required.

System Action: The PCF command server is terminated.

007011E PCF command server stop request failed

Explanation: An attempt to stop the PCF command server failed.

User Response: Check that the system command queue is configured and able to accept command messages. The queue should not be PUT inhibited or full. Also, if MQ security is enabled, check that the stop request was issued by a userid with sufficient authority.

System Action: The PCF command server will not terminate.

007012C PCF command server cannot start command processor

Explanation: The PCF command server received a PCF command on the system command queue but was unable to start the command processor to process the command.

User Response: Check that the command processor is correctly defined to the CICS system. Also check that the CICS system has sufficient resources to start new transactions.

System Action: The PCF command server is terminated.

007013W PCF command server insufficient authority to start processor

Explanation: The PCF command server received a PCF command on the system command queue but was unable to start the command processor to process the command due to an authorization failure.

User Response: Check that the userid running the command server has sufficient authority to start the command processor transaction as any user with authority to put messages to the system command queue. If security is enabled, the command server user must be a surrogate user for users that can put messages to the command queue.

System Action: The PCF command server rejects the command.

007014W PCF command server could not put reply to DLQ

Explanation: The PCF command server attempted to put a PCF reply message to the system dead letter queue and the attempt failed.

User Response: Check that the system dead letter queue is properly configured for the queue manager.

Also check that the dead letter queue is not full or inhibited.

System Action: The PCF reply message is lost.

007015W PCF command server could not be auto-started

Explanation: An attempt to automatically start the PCF command server failed during system initialization.

User Response: Check that the command server is not already running and that the supplied MQ transactions and programs have been installed correctly.

System Action: The PCF command server is not auto-started.

007016W Batch interface could not be auto-started

Explanation: An attempt to automatically start the batch interface failed during system initialization.

User Response: Check that the batch interface is not already running and that the supplied MQ transactions and programs have been installed correctly.

System Action: The batch interface is not auto-started.

007017I PCF command server stopped

Explanation: The PCF command server has been stopped.

User Response: The PCF command server may have stopped due to an error or an operational request. In case of an error, check the system Log for previous error messages.

System Action: PCF command server is no longer available to process PCF commands from the command queue.

007020E PCF command processor could not connect to queue manager

Explanation: An attempt to connect to the local queue manager failed.

User Response: Check that the queue manager is active and is configured to accept a suitable number of concurrent connection requests.

System Action: The PCF command processor attempts to connect to the local queue manager when it is ready to send a PCF response message. The response message is lost, and the result of the original PCF command must be determined manually.

007021W PCF command processor could not open reply queue

Explanation: An attempt to open a locally defined queue for a PCF response message failed.

User Response: Check that the ReplyToQ and ReplyToQMgr fields of the MQMD of the originating PCF command message identify a valid and available queue.

System Action: If the PCF command processor is configured to put undeliverable response message to the system dead letter queue, then an attempt to do so is made. Otherwise, the response message is lost and the result of the original PCF command must be checked manually.

007022W PCF command processor could not send response message

Explanation: An attempt to put a PCF response message to the relevant ReplyToQ failed.

User Response: Check that the relevant queue is available for the receipt of messages.

System Action: If the PCF command processor is configured to put undeliverable response message to the system dead letter queue, then an attempt to do so is made. Otherwise, the response message is lost and the result of the original PCF command must be checked manually.

007023C PCF command processor could not allocate storage

Explanation: An attempt by the PCF command processor to allocate CICS storage failed.

User Response: Check that the relevant CICS partition has sufficient above the line storage to satisfy GETMAIN requests.

System Action: The PCF command processor cannot allocate the necessary storage to generate a PCF response message, and so the response message is lost. The result of the original PCF command must be checked manually.

007024E PCF command processor could not divert to DLQ

Explanation: An attempt by the PCF command processor to put an undeliverable PCF response message to the system dead letter queue failed.

User Response: Check that the system dead letter queue is available and ready to receive messages.

System Action: The PCF response message is lost and the result of the original PCF command must be checked manually.

010000W System started with errors

Explanation: System being initialized but some Queue / Channel definition(s) had error(s).

User Response:

1. Review System Log for prior error messages to identify problem definition.
2. Correct definition(s).
3. Shutdown and then re-initialize system.

System Action: Erroneous Queues / Channels are marked as DISABLED.

010001W System started with file errors

Explanation: System being initialized but some Queues file(s) had error(s).

User Response:

1. Review System Log for prior error messages to identify problem definition.
2. Correct definition(s).
3. Shutdown and then re-initialize system.

System Action: Erroneous Queues are marked DISABLED.

010002W System started with channel errors

Explanation: System being initialized but some channel definition(s) had error(s).

User Response:

1. Review System Log for prior error messages to identify problem definition.
2. Correct definition(s).
3. Shutdown and then re-initialize system.

System Action: Erroneous channel(s) are marked DISABLED.

010003W System started but system changed

Explanation: System being initialized but definitions have been added / deleted while initialization was being performed.

User Response: Do not perform configuration changes while system is being initialized. Shutdown and then re-initialize system.

System Action: If definitions were added then some definition(s) may not be used.

010005I Queue disabled for reorganization

Explanation: The specified queue name has been disabled for a scheduled reorganization of the VSAM file.

User Response: None required.

System Action: None.

010007I Queue enabled by reorganization process

Explanation: The specified queue name has been enabled after the VSAM file has been reorganized.

User Response: None required.

System Action: None.

010020C Global lock error detected by reorganization

Explanation: The global lock table is corrupt for the queue currently being reorganized.

User Response: Stop and restart MQSeries for VSE.

System Action: Reorganization is terminated.

010021W Reorganization target queue is busy

Explanation: The queue is already in use. Automatic Reorganization will stop and retry later.

User Response: You may wish to reschedule reorganization if all the retries have failed.

System Action: Reorganization is terminated, but will be retried, if appropriate.

010022E Redefine of reorganization file failed

Explanation: The IDCAMS delete and redefine of the reorganization dataset failed.

User Response: See CICS console for any error messages from IDCAMS.

System Action: Reorganization is terminated.

010023W Reorganization deleted no records

Explanation: No records have been deleted from the queue file during automatic reorganization.

User Response: None.

System Action: Reorganization continues and the queue file will be reallocated.

010024I Reorganization successful

Explanation: The queue has been reorganized; logically deleted messages have been removed, the underlying VSAM file has been deleted and redefined, and existing messages have been restored.

User Response: None.

System Action: Queue available for processing.

010025E Reorganztion cannot process file with multiple queues

Explanation: During reorganization it has been detected that the associated VSAM file contains records for more than one queue. Automatic reorganization is only possible for files that contain records for a single queue.

User Response: Please ensure only one queue is defined to this queue file. Alternatively you can reorganize this type of file using the offline batch job MQPREORG.

System Action: Reorganization terminates.

010026C Rename of file failed during reorganization

Explanation: The IDCAMS rename for the reorganization file to the queue file failed.

User Response:

1. See CICS console for any error messages from IDCAMS.
2. Manually rename the reorganization file to the queue file.
3. Open the queue file in CICS.
4. Restart the Queue via MQMT.

System Action: Reorganization finishes.

010027W Automatic reorganization already running

Explanation: Reorganization already running. Only one reorganization can run at one time.

User Response: Change the scheduled time for reorganization startup if possible.

System Action: Reorganization is terminated.

010028E Reorganization file not disabled

Explanation: The reorganization file is automatically disabled before it is reallocated, however the attempt to disable the file failed.

User Response:

1. See CICS console for any error messages.
2. Use CICS to manually disable the file before rescheduling the reorganization.

System Action: Reorganization is terminated.

010029E Reorganization file not closed

Explanation: The reorganization file is automatically closed before it is reallocated, however the attempt to close the file failed.

User Response:

1. See CICS console for any error messages
2. Use CICS to manually close the reorganization file before rescheduling the reorganization.

System Action: Reorganization is terminated.

010030E Reorganization file is not defined to CICS

Explanation: The reorganization file must be defined to CICS to automatically reorganize a queue.

User Response: Define the reorganization file to CICS as file MQFREOR.

System Action: Reorganization is terminated.

010031E Reorganization cannot read queue file

Explanation: The queue file must be readable to enable automatic reorganization to take place.

User Response: Ensure the queue file's CICS definition is set to readable.

System Action: Reorganization is terminated.

010032E Reorganization queue file not defined

Explanation: The queue file must be defined to CICS to automatically reorganize a queue.

User Response: Define the queue file to CICS.

System Action: Reorganization is terminated.

010033W Reorganization cannot open queue file

Explanation: An attempt to open the queue file has failed.

User Response:

1. See CICS console for any error messages.
2. Attempt to open the queue file manually in CICS.

System Action: Reorganization continues. See next message.

010034E Reorganization file not open or updatable

Explanation: An attempt to open the reorganization file has failed. The reorganization file must be opened and updatable to enable automatic reorganization to take place.

User Response:

1. See CICS console for any error messages. and resolve the error opening the file.
2. Ensure the reorganization file's CICS definition is set to updatable and can be opened in CICS.

System Action: Reorganization is terminated.

010035W Reorganization cannot close queue file

Explanation: An attempt to close the queue file has failed.

User Response:

1. See CICS console for any error messages and resolve the error closing the file.

System Action: Reorganization ends, and the queue file is started without being reorganized.

010036E Reorganization file not closed

Explanation: An attempt to close the reorganization file has failed.

User Response:

1. See CICS console for any error messages. and resolve the error closing the file.
2. Manually close the file in CICS.
3. Manually rename the IDCAMS reorganization file to the queue file.
4. Open the Queue File in CICS.
5. Restart the queue via MQMT.

System Action: Reorganization ends.

010037E Reorganization file not allocated

Explanation: An attempt to find name of the reorganization file has failed.

User Response: Manually reallocate the reorganization file and reschedule the reorganization.

System Action: Reorganization ends.

010038E Reorganization failed to delete queue file

Explanation: An attempt to delete the queue file has failed during reorganization.

User Response: See CICS console for any error messages from IDCAMS and resolve the error before rescheduling the automatic reorganization.

System Action: Reorganization ends, and the queue file is started without being reorganized.

010039W Deletion of queue failed because queue is not empty

Explanation: An attempt to delete a queue has failed.

User Response: Process queue messages before attempting to delete the queue.

System Action: Queue is not deleted.

100000W System stopped while in use

Explanation: The queue manager was stopped while applications and/or channels were active.

User Response: All applications and channels should be terminated before the system is shutdown.

System Action: System stopped, and active applications and channels are terminated.

100010I System already active

Explanation: An attempt to initialize the queue manager failed because the system is already active.

User Response:

1. Ignore this message, as an initialization attempt was made in error.
2. Stop and restart the queue manager.

System Action: System initialization not performed.

100011W System started with no queues

Explanation: The queue manager has been started, but it has no queue definitions.

User Response: The queue manager can provide no effective services without any queue definitions. Define queues when possible.

System Action: System initialized but inoperative.

100012W System started with too many queues.

Explanation: The queue manager has been started but has more queue definitions than it can process.

User Response: Delete queue definitions as appropriate.

System Action: System initialized with some queue definitions.

100013W System started with too many channels

Explanation: The queue manager has been started but has more channel definitions than it can process.

User Response: Delete channel definitions as appropriate.

System Action: System initialized with some channels.

100090W System started with no queue manager definition

Explanation: The system has been started, but the queue manager definition cannot be found.

User Response: Check that the correct MQ configuration file has been defined to the CICS region. If the queue manager record has been intentionally deleted from the configuration file, a new definition

can be created using the MQMT transaction.

System Action: System initialization is terminated.

100092E Message expiry subsystem failure

Explanation: The message expiry subsystem failed to logically delete an expired message.

User Response: Message expiry occurs when an application attempts to get or browse a message that has expired. At this point, the queue manager attempts to logically delete the message. The queue manager uses several internal transactions and programs to remove expired messages. Ensure that all MQ transactions and programs have been defined and are available to CICS. If maintenance has been recently applied, review any cover letters to ensure special installation instructions have not been missed.

System Action: Message is expired, but is not logically deleted.

101000W Maximum queue depth reached

Explanation: The maximum queue depth for a given queue has been reached, and a further PUT request has been received.

User Response: Perform one of the following :

1. Process this queue either through an application or the queue maintenance facility.
2. Increase the maximum queue depth for the queue.

System Action: The PUT request is terminated and the problem queue is marked as ″MAX″.

101010E Queue concurrent update has occurred

Explanation: Two or more update requests were being received at one time for the same QSN record.

User Response:

1. Review all terminated requests.
2. Re-execute any legitimate requests.

System Action: The first request is served while the subsequent requests, deemed concurrent, are rejected.

101015W Queue not found

Explanation: An attempt to stop/start a queue has failed because the identified queue does not exist.

User Response: Check the queue name matches a queue defined to the queue manager.

System Action: The request is terminated unsuccessfully.

101090I Operation rejected for stopped queue

Explanation: A request has been executed against a STOPPED queue.

User Response: START the problem queue.

System Action: Terminate the request.

101091I Queue disabled and possible operation failure

Explanation: The queue manager flagged a queue disabled due to errors during system initialization, or the queue has been flagged disabled due to an operational failure.

User Response:

1. Examine queue definition and file allocation for problem(s).
2. Check for related error messages in the system log.
3. Correct the queue definition, or resolve associated error messages.

System Action: The problem queue is marked STOPPED.

102090C Queue QSN number limit has been reached

Explanation: Queue messages are internally organized with an incremental numeric key field called a Queue Sequence Number (QSN). Being a signed, 4-byte binary field, the QSN is limited to 2,147,483,650 messages per queue.

User Response: The queue requires reorganization.

1. An offline reorganization can be achieved using the MQ batch utility job MQPREORG.
2. An online reorganization can be achieved using the automatic reorganization feature.

System Action: The PUT request for this queue is rejected.

102091E No space available for PUT request

Explanation: Queue encounters NOSPACE condition for a PUT request.

User Response: Perform one of the following :

1. Do file maintenance on this problem queue such as running the batch job MQPREORG.
2. Execute on/line QUEUE Maintenance to delete messages via ″Delete by Date/time″.

System Action: Terminate the request and mark Queue ″FULL″.

102092E No space available for non-PUT request

Explanation: Queue encounters errors for an UPDATE request, NOSPACE condition occurred.

User Response: Perform one of the following :

1. Do file maintenance on this problem queue such as running the batch job MQPREORG.
2. Execute on/line QUEUE Maintenance to delete messages via ″Delete by Date/time″.

System Action: Terminate the request and mark queue ″FULL″.

104021E Dual queue unavailable

Explanation: Dual destination queue has been STOPPED or was not initialized properly.

User Response: Perform one of the following :

1. Try to re-START the Dual Queue.
2. Examine and fix the Queue and File definition for this queue. Refresh Queue or re-initialize system.

System Action: Marked Dual Queue as ″recovery needed″.

104022E Dual queue file error

Explanation: Dual destination Queue had file error or was not initialized properly.

User Response: Perform one of the following:

1. Try to re-START the Dual Queue.
2. Examine and fix the Queue and File definition for this queue. Refresh Queue or re-initialize system.

System Action: Marked Dual Queue as ″recovery needed″.

104023E Dual queue out of sequence

Explanation: Dual destination Queue does not match Source Queue.

User Response: Examine and fix the Queue and File definition for this queue. Refresh Queue or re-initialize system.

System Action: Marked Dual Queue as ″recovery needed″.

105090E Trigger start by queue start/stop failed

Explanation: MQPSSQ, a subroutine to start / stop a Queue, encounters error to start MQ02, a transaction that handles trigger function.

User Response: Examine CICS tables to fix the problem.

System Action: The request is terminated unsuccessfully.

105091E Trigger initialization failed due to erroneous data

Explanation: MQPAIP2, a program handling trigger function, receives erroneous data and cannot fulfill the request.

User Response: Contact support for MQSeries for VSE.

System Action: The request is terminated unsuccessfully.

109000E Action not authorized

Explanation: NOAUTH condition flagged by CICS when a resource security check has failed, or MQ security is active and an operation was attempted for an unauthorized user.

User Response: Review security mechanism and logs.

System Action: The request is terminated unsuccessfully.

300000E Action not supported

Explanation: Start/stop queue request started with invalid data.

User Response: Review application for call format and data.

System Action: Terminate the request.

300010E Program started incorrectly

Explanation: An MQ module has been started incorrectly.

User Response: Check related MQ documentation to ensure the correct data and data format is being passed to the MQ module.

System Action: MQ module terminates.

300020E Master terminal or derivative incurred map failure

Explanation: MAPFAIL condition raised in Master Terminal panel(s) (MQMT and its derivatives)

User Response: Review PPT for MAP modules (MQM????) and fix the problem.

System Action: Terminate the request.

300030C Queue lock table is full

Explanation: Not enough queue lock entries present to insert a new entry.

User Response: Review application for multiple message retrieval without a SYNCPOINT. If no application problem is present then increase queue lock count to higher value. Note this value is used to

calculate an incore table, so caution should be used.

System Action: Terminate the request.

301000C Expected record is missing

Explanation: An expected message was found missing. This is normally occurs under a Delete request.

User Response: Restart the application.

System Action: Terminate the request.

301010E Duplicate record detected

Explanation: An duplicate message was found. This is normally occurs under a PUT condition.

User Response: Restart the application.

System Action: Terminate the request.

309010E Queue checkpoint record missing

Explanation: A checkpoint of a Queue was requested and no checkpoint record was found on this queue.

User Response: Re-initialize system and restart the application.

System Action: Terminate the request.

400000E Internal operation not recognized

Explanation: A call to an MQ module requests an operation that is not recognized.

User Response: If the MQ module is called by an application, check that the correct data and data format is passed to the MQ module. Otherwise, check that MQSeries has been installed correctly.

System Action: Terminate the request.

400001E Program started with incorrect data length

Explanation: A call to an MQ module passed data that did not match the expected length.

User Response: If the MQ module is called by an application, check that the correct data and data format is passed to the MQ module. Otherwise, check that MQSeries has been installed correctly, or, if maintenance has been recently applied, check that the maintenance has been applied correctly.

System Action: Terminate the request.

400002E Program started with incorrect data

Explanation: A call to an MQ module passed data that was not recognized, or was not in an expected format.

User Response: If the MQ module is called by an application, check that the correct data and data format is passed to the MQ module. Otherwise, check that MQSeries has been installed correctly, or, if maintenance has been recently applied, check that the maintenance has been applied correctly.

System Action: Terminate the request.

400003E Internal call generated bad return code

Explanation: A call to an MQ internal module produced an unexpected return code.

User Response: Check the System Log for related error messages. Also check the VSE/ESA console and CICS logs for possible related error messages.

System Action: Terminate the request.

400010E Internal data corruption

Explanation: Internal MOVE of data has found corrupt data.

User Response: Examine any prior messages for actual problem.

System Action: Terminate the request.

402000C Internal data structure missing

Explanation: An internal data structure was not initialized or has been deleted.

User Response: Check that the queue manager has not been stopped while MQ applications are active. If the queue manager is active, stop and restart the system. If this problem persists, contact MQ/VSE support.

System Action: Terminate the request.

402090C Internal data structure invalid

Explanation: An internal data structure did not contain data as expected.

User Response: For an application program, check that a valid HCONN parameter is passed in all MQI calls. Otherwise, stop and restart the queue manager. If this problem persists, contact MQ/VSE support.

System Action: Terminate the request.

501001E Channel free error

Explanation: (Reserved)

501002E Channel EIB error

Explanation: (Reserved)

501003E Channel closed by remote MCA

Explanation: The remote MCA closed the channel due to an error on the remote system.

User Response: This message indicates the general cause of the channel closure. Error logs on the remote system should be examined for additional information. When the cause has been rectified, the channel can be restarted.

System Action: Fatal error - Communication is terminated.

501004E Channel allocation error

Explanation: (Reserved)

501005E Channel allocate retry error

Explanation: (Reserved)

501006E Channel connect error

Explanation: An attempt to connect a channel failed.

User Response: Check that the channel being connected is correctly defined on both the local and remote MQ systems, and that the channel is defined with the correct channel type.

System Action: Fatal error - Communication is terminated.

501008E Channel send error

Explanation: RECEIVER issued a SEND command and its

User Response:

1. Review System Log or error TD Queue for messages prior to this message. TRM in the error message contains the EIBTRMID which is the principal facility associated with this error. Locate any messages associated with this principal facility.
2. Review the EIBRCODE, EIBRESP, and EIBRESP2 fields in the detail portion of the message. They contain information about the cause of the problem. Refer to the CICS/ESA Application Programming Reference manual for an explanation of these values.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

501009E Receiver responded with error

Explanation: The Sender MCA received an error response from a remote Receiver MCA.

User Response: Review the error reason code to determine the reason of the error response and restart the communication after correction. Also check error logs on remote system for additional information.

System Action: Fatal error - Communication is terminated.

501010E Channel issued invalid response type

Explanation: Unsupported Message Segment Type received.

User Response: Review the Segment type and restart communication without the problem type.

System Action: Fatal error - Communication is terminated.

501011E Channel response MSN error

Explanation: (Reserved)

501012E Channel response indicates fatal error

Explanation: (Reserved)

501013I Channel re-negotiation

Explanation: The Receiver MCA has rejected a channel parameter during activation, and has requested that the remote MCA provide a different parameter value.

User Response: No action is needed unless remote platform can not provide an acceptable channel parameter. If this happens then the conflicting parameter must be changed on this or the remote platform.

System Action: Reject this proposal and continue with channel negotiation.

501014W Channel detected unknown integer encoding

Explanation: The Receiver MCA received an integer encoding parameter of an unknown type.

User Response: Check for error messages on the remote system and take action as necessary.

System Action: Channel communication continues.

501015W Channel received invalid transmission header

Explanation: The Message Channel Agent received data that was not prepended with expected MCA data structures.

User Response: Check for errors on the remote system and take action as necessary.

System Action: Channel communication continues.

501016W Unsupported Coded Character Set ID (CCSID)

Explanation: The Message Channel Agent received a request to use a CCSID that is unknown to the queue manager.

User Response: Define the CCSID to the local queue manager, or configure the remote system to use a known CCSID.

System Action: Remote system can try a different CCSID or terminate the channel.

501017W Channel received invalid message segment header

Explanation: The Receiver MCA received data that did not have the embedded data structures that it expected.

User Response: Check for errors on the remote system and take action as necessary.

System Action: Channel communication is terminated.

501018W Channel received invalid transmission queue header

Explanation: The Receiver MCA received data that did not have the embedded data structures that it expected.

User Response: Check for errors on the remote system and take action as necessary.

System Action: Channel communication is terminated.

501019W Channel initiation failed

Explanation: The Receiver MCA could not initiate a channel.

User Response: Check for errors on the remote system and take action as necessary.

System Action: Channel communication is terminated.

501020W Unsupported FAP level

Explanation: In establishing a channel connection, a remote MQ system is attempting to use a protocol level that is not recognized by the local queue manager.

User Response: The channel activation process should negotiate a protocol level that is acceptable to both queue managers.

System Action: Channel activation continues.

501021W Message size too large

Explanation: During channel negotiation, a maximum message size was requested, but was too large for one of the queue managers.

User Response: If channel activation fails, change the channel definition to use a smaller maximum message size.

System Action: Channel negotiation continues.

501022W Message wrap mismatch

Explanation: During channel negotiation, a wrap sequence number was requested, but did not match the wrap value in the associated channel definition.

User Response: If channel activation fails, ensure that sender and receiver channel pairs use the same wrap sequence value.

System Action: Channel negotiation continues.

501023E Undeliverable message lost

Explanation: An attempt to place an undeliverable message to the system dead letter queue failed because the dead letter queue was unavailable or the queue manager was inactive.

User Response: Check that the system dead letter queue is correctly defined and available. Determine the intended target queue for the message from the channel status, and verify that the target queue is correct and available. Recreate the lost message if necessary.

System Action: Process is terminated.

501024E Queue manager unavailable for communication

Explanation: A channel cannot be established because one of the queue managers is unavailable.

User Response:

1. Check local queue manager and if necessary initialize using MQIT or MQMT.
2. Check the queue manager is active on the remote system.

System Action: Process or communication is terminated.

501025E Unknown inbound channel name

Explanation: The communication cannot be established because the channel name received from the remote system is not defined locally.

User Response: Check the channel name to see if it is correct. Define this in the local definitions or correct the remote system as necessary.

System Action: This communication session is terminated.

501026E Unknown outbound channel name

Explanation: The channel specified is not available on the remote platform.

User Response: Check on the remote system to see why the channel is not available and define it if necessary.

System Action: This communication session is terminated.

501027E Channel busy or stopped

Explanation: An attempt was made to use a channel that is already in use, or has been stopped.

User Response:

1. Retry the channel later.
2. Start the channel for immediate use.

System Action: Fatal error - Communication is terminated.

501028W Channel re-synchronization error

Explanation: During channel activation, the Receiver MCA detected that the requested message sequence number did not match the expected value.

User Response: If the channel will not start, verify that expected messages have been sent/received, and manually reset the MSN for both the sender and the receiver channels.

System Action: Channel activation continues.

501029E Channel unavailable for use

Explanation: An attempt was made to use a channel that is currently unavailable.

User Response: Check the Enable flag in the channel definition and ensure that it is set to 'Y'. Enable the channel or disable the transmission to prevent unnecessary log messages.

System Action: The MCA is terminated without further action.

501030E Transmission length error

Explanation: The Receiver MCA has detected a mismatch between an explicit data length and the length of data received over an active connection.

1. The length of the application portion of the message specified in the header exceeds the maximum length defined for this channel.
2. The length of the application portion of the message received is not equal to the length specified in the header.

User Response: For reason #1:

1. Review the MAX STA LEN and the MES LEN in the detail portion of the message.
2. Check the configuration of the Receiver channel to insure the maximum message size is set up correctly.
3. Check the configuration of the Sender.
4. Reconfigure if necessary and restart communication.

For reason #2:

1. Review the HEAD MES LEN and the REC LEN in the detail portion of the message.
2. Proper running should preclude this occurrence. Investigate sender/server process for program error.
3. Correct problem and restart communication.

System Action: Fatal error - Communication is terminated.

501031W Messages per batch too large

Explanation: During channel negotiation, a messages per batch value was requested, but was too large for one of the queue managers.

User Response: If channel activation fails, change the channel definition to use a smaller batch size.

System Action: Channel negotiation continues.

501032W Maximum transmission size too large

Explanation: During channel negotiation, a maximum transmission size was requested, but was too large for one of the queue managers.

User Response: If channel activation fails, change the channel definition to use a smaller transmission size.

System Action: Channel negotiation continues.

501050I Message sequence number has been reset

Explanation: During channel activation, the Receiver MCA has detected that the message sequence number has been manually reset.

User Response: If the channel does not start, manually reset the channel MSN to match the remote system. Care should be taken that expected messages have been transmitted, and transmitted messages are not duplicated by being resent.

System Action: Channel negotiation continues if the MSN is valid.

501060E Unexpected or invalid channel security exit exchange

Explanation: During channel activation, a channel security exit has been activated that returned an error return code or did not receive an expected security exchange from the remote system.

User Response: Ensure that the correct exit programs are configured to run at either end of the channel. Ensure that the security requirements for the channel are met before restarting the channel.

System Action: Channel is terminated.

501061E Channel security exit program missing or unavailable

Explanation: During channel activation, the initialization of a channel security exit program failed because it was unavailable to the CICS region.

User Response: Ensure that the correct exit program is specified in the channel definition. Check that the program is defined to CICS and enabled.

System Action: Channel is terminated.

501062E Unexpected or invalid channel message exit response

Explanation: During channel processing, a channel message exit returned an invalid response, or a request to terminate the channel.

User Response: Ensure that the correct exit programs are configured to run at either end of the channel. Determine the reason for the message exit response and correct.

System Action: Channel is terminated.

501063E Channel message exit program missing or unavailable

Explanation: During channel activation, the initialization of a channel message exit program failed because it was unavailable to the CICS region.

User Response: Ensure that the correct exit program is specified in the channel definition. Check that the program is defined to CICS and enabled.

System Action: Channel is terminated.

501064E Unexpected or invalid channel send exit response

Explanation: During channel processing, a channel send exit returned an invalid response, or a request to terminate the channel.

User Response: Ensure that the correct exit programs are configured to run at either end of the channel. Determine the reason for the send exit response and correct.

System Action: Channel is terminated.

501065E Channel send exit program missing or unavailable

Explanation: During channel activation, the initialization of a channel send exit program failed because it was unavailable to the CICS region.

User Response: Ensure that the correct exit program is specified in the channel definition. Check that the program is defined to CICS and enabled.

System Action: Channel is terminated.

501066E Unexpected or invalid channel receive exit response

Explanation: During channel processing, a channel receive exit returned an invalid response, or a request to terminate the channel.

User Response: Ensure that the correct exit programs are configured to run at either end of the channel. Determine the reason for the receive exit response and correct.

System Action: Channel is terminated.

501067E Channel receive exit program missing or unavailable

Explanation: During channel activation, the initialization of a channel receive exit program failed because it was unavailable to the CICS region.

User Response: Ensure that the correct exit program is specified in the channel definition. Check that the program is defined to CICS and enabled.

System Action: Channel is terminated.

501070E Channel exit returned invalid data length

Explanation: A channel exit returned a data length greater than the maximum transmission length.

User Response: Check channel exit logic and ensure that the data length returned by the exit does not exceed the maximum transmission length.

System Action: Channel is terminated.

501071E Channel exit returned bad exit buffer address

Explanation: A channel exit returned a response code to use an exit buffer, but the exit buffer address was not valid.

User Response: Check channel exit logic and ensure that the exit buffer address is valid when the response code indicates its use is required.

System Action: Channel is terminated.

501072E Channel exit modified MCA transmission header

Explanation: A send or receive channel exit modified the first 8 bytes of the transmission data.

User Response: Check channel exit logic and ensure that the exit does not modify the first 8 bytes of the transmission data.

System Action: Channel is terminated.

800000E Unexpected CICS condition

Explanation: An MQ module trapped a generically handled error condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason.

System Action: Terminate the request.

800010E Unexpected CICS INVREQ condition raised

Explanation: An MQ module trapped a CICS INVREQ condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason.

System Action: Terminate the request.

800011C Unexpected CICS ILLOGIC condition raised

Explanation: An MQ module trapped a CICS ILLOGIC condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason.

System Action: Terminate the request.

800090E Checkpoint processing failed

Explanation: A general error occurred while processing the checkpoint record of a Queue file.

User Response: Use LISTCAT to review the VSAM file containing this Queue file.

System Action: Terminate the request.

800099E CICS abnormal end condition

Explanation: An MQ module trapped a CICS ABEND condition.

User Response: Perform normal CICS application abend analysis to determine the cause of the abend.

System Action: Terminate the request.

801012E Unexpected CICS NOTOPEN condition raised

Explanation: An MQ module trapped a CICS NOTOPEN condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason.

System Action: Terminate the request.

801019E Unexpected CICS DISABLE condition raised

Explanation: An MQ module trapped a CICS DISABLE condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason.

System Action: Terminate the request.

802000C Unexpected CICS NOSTG condition raised

Explanation: An MQ module trapped a CICS NOSTG condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason, and

1. check that you have not overestimated the maximum size of messages in queue and channel definitions
2. if you have large message sizes then increase the amount of 31-bit storage in the partition that CICS is running

System Action: Terminate the request.

803001C Unexpected CICS LENGERR condition raised

Explanation: An MQ module trapped a CICS LENGERR condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that recent maintenance has been applied correctly.

System Action: Terminate the request.

808000C Unexpected CICS MAPFAIL condition raised

Explanation: An MQ module trapped a CICS MAPFAIL condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that

recent maintenance has been applied correctly.

System Action: Terminate the request.

809000C Unexpected CICS PGMIDERR condition raised

Explanation: An MQ module trapped a CICS PGMIDERR condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that recent maintenance has been applied correctly. If the error is caused by a trigger event, check that the trigger program is defined to CICS.

System Action: Terminate the request.

809010C Unexpected CICS FILEID condition raised

Explanation: An MQ module trapped a CICS FILEID condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that recent maintenance has been applied correctly.

System Action: Terminate the request.

809011C Unexpected CICS FILENOTFOUND condition raised

Explanation: An MQ module trapped a CICS FILENOTFOUND condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that recent maintenance has been applied correctly.

System Action: Terminate the request.

809012C Unexpected CICS IOERR condition raised

Explanation: An MQ module trapped a CICS IOERR condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that recent maintenance has been applied correctly.

System Action: Terminate the request.

809050C Unexpected CICS TRANIDERR condition raised

Explanation: An MQ module trapped a CICS TRANIDERR condition.

User Response: Produce a CICS auxiliary trace to determine the CICS command failure and the error reason. Check that MQ is installed correctly and that recent maintenance has been applied correctly. If the error is caused by a trigger event, check that the trigger transaction and its associated program are defined to CICS.

System Action: Terminate the request.

900000C Queue manager environment missing or invalid

Explanation: An attempt to use queue manager services was made without the system environment being established.

User Response: Run transaction MQSE to establish the queue manager environment. If the environment exists, check that MQ is installed correctly and that recent maintenance has been applied correctly.

System Action: Terminate the request.

All messages starting with 600000 are critical messages displayed on the CICS terminals from which MQSeries Administrator Dialogs (MQMT) have been started.

These messages indicate failures in the MQSeries code itself. Each message number is followed by the program name in which the failure occurred.

If after checking, and making any possible corrections, the problem persists, report this to your IBM support organization. You should include the message number, together with the names of the modules and any other information.

600001C Prog: xxxxxxxx Error detected. Contact Support.

Explanation: CICS has detected an error condition not handled by a specific routine.

Operator action: Report to IBM

System action: The dialog was ended.

600005C Prog: xxxxxxxx ABEND Code zzzz Contact Support.

Explanation: The program ended due to a CICS problem and the ABEND code *zzzz* was returned to a HANDLE ABEND routine.

Operator action: Report to IBM

System action: The dialog was ended.

600007C **Prog: xxxxxxxx File: yyyyyyy Not Found. Contact Support.**

Explanation: A request has been issued against the file *yyyyyyyy*, but it was not defined in the FCT

Operator action: Contact your system administrator and check whether all MQSeries files were defined in the CICS File Control Table (FCT), and physically allocated by VSAM.

System action: The dialog was ended.

600009C **Prog: xxxxxxxx File: yyyyyyy DISABLED. Contact Support.**

Explanation: CICS tried to access the file *yyyyyyy* which was not enabled.

Operator action: Use "CEMT S DATA" to set the file ENABLED. If the DISABLED status persists, check with the System Administrator.

System action: The dialog was ended.

600011C **Prog: xxxxxxxx File: yyyyyyy ILLOGIC error. Contact Support.**

Explanation: Usually this is related to file input and output. This condition is returned by CICS when the error does not fall within one of the other CICS response categories.

Operator action: Report to IBM

System action: The dialog was ended.

600017C **Prog: xxxxxxxx File: yyyyyyy I/O error. Contact Support.**

Explanation: Normally this is due to hardware errors.

Operator action: Check the System console for more details.

System action: The dialog was ended.

600019C **Prog: xxxxxxxx File: yyyyyyy Record not found. Contact Support.**

Explanation: The program tried to read a record but the request failed.

Operator action: Report to IBM.

System action: The dialog was ended.

600021C **Prog: xxxxxxxx File: yyyyyyy is not open. Contact Support.**

Explanation: CICS tried to access a file which had not been opened, and was unable to open it. This can happen when the file is already in use by another partition.

Operator action: Use "CEMT I DATA" and try to open it manually.

System action: The dialog was ended.

600023C **Prog: xxxxxxxx INVREQ error Contact Support.**

Explanation: A request was received by CICS and cannot be processed for various reasons.

Operator action: Report to IBM

System action: The dialog was ended.

600025C **Prog: xxxxxxxx MAPFAIL error Contact Support.**

Explanation: CICS was unable to display a BMS map on the terminal.

Operator action: Report to IBM

System action: The dialog was ended.

600027C **Prog: xxxxxxxx TRANSID error Contact Support.**

Explanation: MQSeries tried to initiate a transaction, but this transaction was not found in the CICS tables.

Operator action: This is probably an installation error. Check whether the MQSeries group has been correctly installed in the DFHCSD file, and activated. Use CEMT I TRAN(MQ*) to verify this. If everything appears to be correct, report the problem to IBM.

System action: The dialog was ended.

800000E **CICS ERROR CONDITION REACHED**

Explanation: ERROR condition of CICS occurred.

Function: General (CICS Interface)

Operator action: Investigate the error.

System action: End the request.

800010E **INVALID REQUEST CONDITION**

Explanation: INVREQ (Invalid Request) condition of CICS reached.

Function: General (CICS Interface)

Operator action: Investigate the error.

System action: End the request.

800011C **ILLOGIC CONDITION**

Explanation: ILLOGIC condition of CICS occurred.

Function: General (CICS Interface)

Operator action: Investigate the error.

System action: End the request.

800090E ERROR CONDITION DURING CHECKPOINT PROCESSING

Explanation: A general error occurred while processing the checkpoint record of a queue file.

Function: General (I/O modules MQPQUE1 and MQPQUE2)

Operator action: Use LISTCAT to review the VSAM file containing this queue file.

System action: End the request.

800099E CICS ABEND CONDITION REACHED

Explanation: ABEND condition of CICS occurred.

Function: General (CICS Interface)

Operator action: Investigate the error.

System action: End the request.

801012E FILE NOTOPEN CONDITION

Explanation: A CICS file entry has been CLOSED.

Function: General (CICS Interface)

Operator action: Check install of CICS table.

System action: End the request.

801019E DISABLE CONDITION

Explanation: A CICS table entry has been DISABLED.

Function: General (CICS Interface)

Operator action: Check install of CICS table.

System action: End the request.

802000C NO STORAGE CONDITION

Explanation: A CICS storage is not available.

Function: General (CICS Interface)

Operator action: Check that the user task has not freed storage.

System action: End the request.

803001C LENGTH ERROR CONDITION

Explanation: A record was larger than expected.

Function: General (CICS Interface)

Operator action: Check that the product has been correctly installed.

System action: End the request.

808000C MAPFAIL CONDITION

Explanation: A CICS transaction is missing.

Function: General (CICS Interface)

Operator action: Check install of CICS PPT table for maps.

System action: End the request.

809000C PGMIDERR CONDITION

Explanation: A CICS program id is missing.

Function: General (CICS Interface)

Operator action: Check install of CICS PPT table.

System action: End the request.

809010C FILEID CONDITION

Explanation: No file was available to process.

Function: General (CICS Interface)

Operator action: Check install for CICS FCT table.

System action: End the request.

809011C NOFILE CONDITION

Explanation: No file was available to process.

Function: General (CICS Interface)

Operator action: Check install for CICS FCT table.

System action: End the request.

809012C IO ERROR CONDITION

Explanation: A CICS I/O error has occurred.

Function: General (CICS Interface)

Operator action: Check CICS log and EIB codes.

System action: End the request.

809050C TRANIDERR CONDITION

Explanation: A CICS transaction is missing.

Function: General (CICS Interface)

Operator action: Check install of CICS PCT table.

System action: End the request.

900000C ENVIRONMENT RECORD

Explanation: Setup of Environment has not been performed.

Function: Set up system

Operator action: Process Transaction MQSE to setup Environment.

System action: End the request.

Console Messages

The following messages may be generated by MQSeries and displayed on the VSE/ESA console during normal operation.

MQI0010I **MQSeries for VSE/ESA environment initializing.**

MQI0011E **File MQFCNFG not defined in CICS.**

MQI0013E **MQFCNFG disabled. Environment not initialized.**

MQI0014E **MQFCNFG not upgraded with 2.1.2 SYSIN data.**

MQI0015E **MQFCNFG VSAM ERROR**

MQI0020I **MQSeries for VSE/ESA environment initialized.**

MQI0030I **MQSeries for VSE/ESA system starting.**

MQI0032W **MQ/Security unavailable with CICS release.**

MQI0035I **MQSeries for VSE/ESA licensed support for nnnn clients.**

MQI0037W **CICS SIT parameters may restrict number of MQ/VSE MCA or MQ/VSE Client connections.**

MQI0040I **MQSeries for VSE/ESA system started.**

MQI0090I **MQSeries for VSE/ESA system stopped.**

MQI0091E **Error has occurred during shutdown.**

Batch Interface Console Messages

The following message may be generated by the MQSeries batch interface, or MQSeries batch applications. For all messages, the symbol 'mqintfid' is replaced by the batch interface identifier of the queue manager.

Messages generated from MQSeries batch applications:

MQB0001E **MQS Batch Interface (mqintfid) XPCC error RETC(rc) REAS(re)**

Explanation: An XPCC operation failed when an MQI batch application issued an MQI call. The XPCC call issued returned code 'rc' and reason code 're'.

Operator Response: Check that the MQSeries batch interface is running in the CICS region that hosts MQSeries. Review VSE/ESA System Macro documentation for descriptions of the XPCC return and reason codes.

MQB0008E **MQS Batch Interface (mqintfid) XPCC SENDR failed**

Explanation: An XPCC SENDR operation failed. This message should be accompanied by message MQB0001E which describes the XPCC return and reasons codes.

Operator Response: Examine MQB0001E message and follow operator action instructions.

MQB0009E **MQS Batch Interface (mqintfid) XPCC CONNECT failed**

Explanation: An XPCC CONNECT operation failed. This message should be accompanied by message MQB0001E which describes the XPCC return and reasons codes.

Operator Response: Examine MQB0001E message and follow operator action instructions.

MQB0010E **MQS Batch Interface (mqintfid) XPCC IDENTIFY failed**

Explanation: An XPCC IDENTIFY operation failed. This message should be accompanied by message MQB0001E which describes the XPCC return and reasons codes.

Operator Response: Examine MQB0001E message and follow operator action instructions.

MQB0013E **MQS Batch Interface (mqintfid) GETVIS failed**

Explanation: An attempt to obtain GETVIS storage failed.

Operator Response: Increase the available GETVIS storage for the batch partition.

Messages generated by the MQSeries batch interface:

MQI0100I MQS Batch Interface (mqintfid) started.

Explanation: The MQSeries batch interface has started.

Operator Response: None.

MQI0101E MQS Batch Interface (mqintfid) GETMAIN failed

Explanation: The MQSeries batch interface failed to obtain sufficient GETMAIN or GETVIS storage.

Operator Response: Increase the storage resources available to the CICS region.

MQI0102I MQS Batch Interface (mqintfid) stop requested

Explanation: A request to stop the MQSeries batch interface has been registered.

Operator Response: None.

MQI0103E MQS Batch Interface (mqintfid) userid error

Explanation: The MQSeries batch interface cannot start a mirror transaction (MQBX) for a batch program because the user associated with the interface is not a surrogate for the batch user, or the user is invalid.

Operator Response: Check that the batch job includes a valid // ID card that identifies a valid userid and password, and that the interface user is a surrogate for that userid.

MQI0104I MQS Batch Interface (mqintfid) ended

Explanation: The MQSeries batch interface has stopped.

Operator Response: None.

MQI0105E MQS Batch Interface (mqintfid) abending

Explanation: The MQSeries batch interface has encountered an abend condition and is terminating.

Operator Response: Wait for the interface to abend, check accompanying console messages. Examine relevant dumps.

MQI0107E MQS Batch Interface (mqintfid) init failed

Explanation: The MQSeries batch interface could not establish an XPCC identity for batch connections.

Operator Response: Check that the queue manager is configured with a valid batch interface identifier.

MQI0108I MQS Batch Interface (mqintfid) system inactive

Explanation: An attempt to start the MQSeries batch interface was made when the MQSeries queue manager was inactive.

Operator Response: Start the MQSeries queue manager before starting the batch interface.

MQI0109I MQS Batch Interface (mqintfid) not started

Explanation: An attempt to stop the MQSeries batch interface was made when the interface was not active.

Operator Response: None.

Automatic reorganization console messages

Messages prefixed by "MQPVSAM:" are only displayed when an error occurs. Refer to the MQ SYSTEM.LOG for more details.

The MQ/VSE automatic reorganization feature can also generate the follow message:

MQPIDCMS Insufficient below line GETVIS for reorg

Explanation: There is not enough GETVIS-24 to perform the automatic reorganization.

Operator action: Refer to CICS system programmer to allocate more GETVIS-24 to this partition.

System action: The automatic reorganization is not performed.

Appendix H. Security implementation

This appendix provides a sample security configuration. The sample includes configuration for:

- MQSeries datasets
- MQSeries transactions
- MQSeries system users
- Application users
- Connections
- Queues
- Batch users
- Clients
- Commands
- Command resources
- MQSeries startup
- MQSeries shutdown

The examples in this appendix use CA-Top Secret® as the External Security Manager (ESM).

Note: The examples in this appendix are only a sample security configuration, and are not intended to define how you should secure your VSE/ESA or CICS TS systems.

Before you install

Before installation, you need to make several decisions regarding security. The first is whether to install security or not. If you do want to install security, as this appendix assumes, you need to:

- modify the MQCICDCT sample JCL
- modify the SYSIN installation parameter file

These two steps are described in Chapter 2, “Installation,” on page 7.

The MQCICDCT.Z sample DCT definition file must be modified if you want a user other than the CICS default user to log messages to the SYSTEM.LOG or manage message expiry. Remember that logging messages requires CONNECT and QUEUE authority, and message expiry involves CONNECT and authority to put messages to reply queues.

In addition, the generation of instrumentation events requires that the IE processor transaction triggered by requests arriving on the MQIE transient data queue require CONNECT and OPEN access for the event queues.

This example uses a user other than the CICS default user. The DCT definitions should be changed as follows:

```
MQER     DFHDCT TYPE=INTRA,
                RSL=PUBLIC,
                DESTID=MQER,
                DESTFAC=FILE,
                USERID=MQADMIN,  <--- Note insertion
                TRANSID=MQER,
                TRIGLEV=1
```

© Copyright IBM Corp. 1995, 2004

```
MQXP     DFHDCT TYPE=INTRA,
                RSL=PUBLIC,
                DESTID=MQXP,
                DESTFAC=FILE,
                USERID=MQADMIN,  <---Note insertion
                TRANSID=MQXP,
                TRIGLEV=1
MQIE     DFHDCT TYPE=INTRA,
                RSL=PUBLIC,
                DESTID=MQIE,
                DESTFAC=FILE,
                USERID=MQADMIN,   <---Note insertion
                TRANSID=MQIE,
                TRIGLEV=1
```

The SYSIN.Z installation configuration file contains default settings for security, where the default is DISABLED. You must change this default to ENABLED before installation. However, you cannot do this until you have installed the MQSeries library from tape.

The relevant entries in the SYSIN parameter file follow the QMDEF heading and should be changed from:

```
QM-STATUS-SECURITY       DISABLED
QM-AUDIT-SECURITY        DISABLED
```

To:

```
QM-STATUS-SECURITY       ENABLED
QM-AUDIT-SECURITY        ENABLED
```

Note that the AUDIT parameter is not implemented in Version 2.1.2, but is reserved for future expansion. It is enabled in this example for consistency.

External security manager configuration

For MQSeries security to work, certain facilities must be available with your ESM. For CA-Top Secret®, you need to ensure the following settings exist in the system parameter file:

```
FACILITY(CICSPROD=MODE=FAIL)
FACILITY(CICSPROD=FACMATRX=YES)
FACILITY(CICSPROD=EXTSEC=YES)
FACILITY(CICSPROD=XFCT=YES)
FACILITY(CICSPROD=XDEF)
FACILITY(CICSPROD=XUSER=YES)
FACILITY(CICSPROD=RES)
```

These parameters assume that the facility of the MQSeries CICS region will be CICSPROD. If you are using CICSTEST, or a different facility, you should make the appropriate changes to the parameter file.

The `CICSPROD=RES` parameter ensures the standard MQSeries classes are available. If you change the parameter file, you also need to stop and restart your CA-Top Secret® system.

System and application users

This example uses the following system users:

CICSP1 CICS region user

CICSP1DF CICS default user

MQM	Owner of all MQSeries resources
MQADMIN	MQSeries Startup and Administrative user

The example also uses the following application users:

JOHNS	Application user
JANED	Application user
SHELLYS	Client user
STEVEJ	Batch user

To create these users, you might use the following TSS commands:

```
--  CICS & MQ Departments
TSS CREATE(CICSP1G)  NAME('CICSP1 GROUP')         TYPE(DEPARTMENT)
TSS CREATE(MQ)       NAME('MQSERIES 2.1.2 GROUP') TYPE(DEPARTMENT)

--  CICS System Users
TSS CREATE(CICSP1)   NAME('CICSP1 REGION') TYPE(USER) FAC(BATCH)  +
                     DEPT(CICSP1G) PAS(P1CICS,0)                  +
                     MASTFAC(CICSPROD) NORESCHK NOLCFCHK NODSNCHK
TSS CREATE(CICSP1DF) NAME('CICSP1 DEFAULT USER')  TYPE(USER)      +
                     DEPT(CICSP1G) PAS(NOPW,0) FAC(CICSPROD)

--  MQ System Users
TSS CREATE(MQM)      NAME('MQM OWNER')            TYPE(USER)      +
                     DEPT(MQ) PAS(MQSER1ES,0) FAC(CICSPROD)
TSS CREATE(MQADMIN)  NAME('MQSERIES ADMIN USER')  TYPE(USER)      +
                     DEPT(CICSP1G) PAS(ADM1N,0) FAC(CICSPROD)

--  MQ Application Users
TSS CREATE(JOHNS)    NAME('JOHN SMITH')           TYPE(USER)      +
                     DEPT(MQ) PAS(SM1TH,0)   FAC(CICSPROD)
TSS CREATE(JANED)    NAME('JANE DOE')             TYPE(USER)      +
                     DEPT(MQ) PAS(D0E,0)     FAC(CICSPROD)
TSS CREATE(SHELLYS)  NAME('SHELLY SIMPSON')       TYPE(USER)      +
                     DEPT(MQ) PAS(SIMPS0N,0) FAC(CICSPROD)
TSS CREATE(STEVEJ)   NAME('STEVEN JONES')         TYPE(USER)      +
                     DEPT(MQ) PAS(J0NES,0)   FAC(BATCH,CICSPROD)
```

MQSeries datasets

Before you start up your CICS and MQSeries systems, you should consider MQSeries dataset security. There is little point in protecting MQSeries resources, but then allowing unrestricted access to the MQSeries datasets in which such objects reside.

To protect your datasets in CICS, assuming they use the default names provided with the sample JCL, you could use the following TSS commands:

```
-- Assign ownership
TSS ADD(MQM) FCT(MQFCNFG)
TSS ADD(MQM) FCT(MQFLOG)
TSS ADD(MQM) FCT(MQFMON)
TSS ADD(MQM) FCT(MQFERR)
TSS ADD(MQM) FCT(MQFREOR)
TSS ADD(MQM) FCT(MQFSSET)
TSS ADD(MQM) FCT(MQFI001)
TSS ADD(MQM) FCT(MQFI002)
TSS ADD(MQM) FCT(MQFI003)
TSS ADD(MQM) FCT(MQFO001)
TSS ADD(MQM) FCT(MQFO002)
TSS ADD(MQM) FCT(MQFO003)
```

```
-- Assign permissions to Admin user
TSS PER(MQADMIN) FCT(MQFCNFG) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFLOG)  ACC(ALL)
TSS PER(MQADMIN) FCT(MQFMON)  ACC(ALL)
TSS PER(MQADMIN) FCT(MQFERR)  ACC(ALL)
TSS PER(MQADMIN) FCT(MQFREOR) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFSSET) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFI001) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFI002) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFI003) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFO001) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFO002) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFO003) ACC(ALL)
```

If the Programmable Command Formats (PCF) and MQSeries Command (MQSC) features are required, the following datasets should also be protected:

```
TSS PER(MQADMIN) FCT(MQFACMD) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFARPY) ACC(ALL)
```

Similarly, if the Instrumentation Events feature is required, the following datasets should also be protected:

```
TSS PER(MQADMIN) FCT(MQFIEQE) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFIECE) ACC(ALL)
TSS PER(MQADMIN) FCT(MQFIEPE) ACC(ALL)
```

At this point, because the facilities matrix for CICSPROD has `XFCT= YES`, no users other than MQM and MQADMIN have access to the MQSeries datasets under CICS. Appropriate permissions for other users are described later.

You should also protect your datasets outside CICS. To do this, you might use the following:

```
TSS ADD(MQM) DSN(MQSERIES)
```

This command establishes generic ownership of all datasets that are prefixed with MQSERIES.

Protecting transactions

The `CICSPROD=XDEF` facility matrix setting ensures that users cannot run transactions without explicit permission. In this example, you grant full access to the MQADMIN user, and restricted access to application users:

```
TSS ADD(MQADMIN) TRANS(CICSPROD,MQ(G),TST(G))
```

By default, with CA-Top Secret®, transactions beginning with TS are in the protection bypass list. Because MQSeries uses TST1, TST2, and TST3 transactions, to introduce protection, you should issue the following command:

```
TSS MODIFY FAC(CICSPROD=PROTADD(TRANID=TST))
```

You should also consider protecting programs. In this example, restricting access to transactions in CICS should be sufficient, and the protection of programs is omitted.

Regardless of whether the MQSeries transactions are protected, if security is active, the use of the transactions should be further restricted by command and command resource security.

Command and command resource security not only restrict the use of the MQSeries transactions, but also PCF and MQSC commands.

Resource ownership

The ESM classes that are relevant to MQSeries for VSE/ESA are:
- MQADMIN
- MQCONN
- MQQUEUE
- MQCMDS

Resources defined to these classes must first have ownership. For this example, all such resources will be owned by user MQM. You can assign ownership as follows:

```
TSS ADD(MQM) MQADMIN(VSE.QM1)
TSS ADD(MQM) MQCONN(VSE.QM1)
TSS ADD(MQM) MQQUEUE(VSE.QM1)
TSS ADD(MQM) MQCMDS(VSE.QM1)
```

Because these classes are generic, all resources prefixed VSE.QM1 are owned automatically by user MQM. At this point, no specific resources have been defined. Appropriate user permissions for class resources are described later.

Resource protection

At this point, MQSeries datasets and transactions are protected. You are now ready to grant users access to MQSeries resources. These include:
- Connections
- Queues

This example makes the following assumptions:
- All application users have authority to connect to the MQSeries queue manager.
- Queue TEST.Q1 is defined in file MQFI001.
- Queue TEST.Q2 is defined in file MQFI002.
- Queue TEST.Q3 is defined in file MQFI003.
- Application user JOHNS requires read/write authority to all three application queues.
- Application user JANED requires read/write authority to TEST.Q1 and TEST.Q2.
- Application user SHELLYS is a client user and requires authority to read/write to TEST.Q3.
- Application user STEVEJ is a batch user and requires authority to browse TEST.Q3.

To grant authority to all users to connect to the MQSeries queue manager, issue the following commands:

```
TSS PER(MQADMIN) MQCONN(VSE.QM1.CICS) ACC(READ)

TSS PER(JOHNS)   MQCONN(VSE.QM1.CICS) ACC(READ)
TSS PER(JANED)   MQCONN(VSE.QM1.CICS) ACC(READ)
TSS PER(SHELLYS) MQCONN(VSE.QM1.CICS) ACC(READ)
TSS PER(STEVEJ)  MQCONN(VSE.QM1.CICS) ACC(READ)
```

Note that the MQADMIN user is also the user assigned to the MQER DCT entry. This user must have CONNECT authority and must be able to write to the SYSTEM.LOG queue.

To grant authority to each user to access specific queues, using the assumptions listed earlier, issue the following commands:

```
TSS PER(MQADMIN) MQQUEUE(VSE.QM1) ACC(ALL)

TSS PER(JOHNS)   MQQUEUE(VSE.QM1.TEST.Q1) ACC(READ,UPDATE)
TSS PER(JOHNS)   MQQUEUE(VSE.QM1.TEST.Q2) ACC(READ,UPDATE)
TSS PER(JOHNS)   MQQUEUE(VSE.QM1.TEST.Q3) ACC(READ,UPDATE)

TSS PER(JANED)   MQQUEUE(VSE.QM1.TEST.Q1) ACC(READ,UPDATE)
TSS PER(JANED)   MQQUEUE(VSE.QM1.TEST.Q2) ACC(READ,UPDATE)

TSS PER(SHELLYS) MQQUEUE(VSE.QM1.TEST.Q3) ACC(READ,UPDATE)

TSS PER(STEVEJ)  MQQUEUE(VSE.QM1.TEST.Q3) ACC(READ)
```

Access to underlying MQSeries datasets must also be granted. Following the listed assumptions, you need to issue the following commands:

```
TSS PER(JOHNS)   FCT(MQFI001) ACC(INQUIRE,READ,WRITE)
TSS PER(JOHNS)   FCT(MQFI002) ACC(INQUIRE,READ,WRITE)
TSS PER(JOHNS)   FCT(MQFI003) ACC(INQUIRE,READ,WRITE)

TSS PER(JANED)   FCT(MQFI001) ACC(INQUIRE,READ,WRITE)
TSS PER(JANED)   FCT(MQFI002) ACC(INQUIRE,READ,WRITE)

TSS PER(SHELLYS) FCT(MQFI003) ACC(INQUIRE,READ,WRITE)

TSS PER(STEVEJ)  FCT(MQFI003) ACC(INQUIRE,READ)
```

For testing purposes, you can use the TST2 transaction, which allows users to read and write messages to queues. To allow users to use this transaction, issue the following commands:

```
TSS ADD(JOHNS)   TRANS(CICSPROD,TST2)
TSS ADD(JANED)   TRANS(CICSPROD,TST2)
```

Note that SHELLYS and STEVEJ do not need the TST2 transaction. SHELLYS is a client user and can issue MQI calls directly from a remote MQI client program. STEVEJ is a batch user and similarly can issue MQI calls from a batch partition.

Batch user permissions

Batch users identify themselves to the External Security Manager via the // ID card. For example:

```
// ID USER=STEVEJ,PWD=J0NES
```

MQSeries security uses the user name from the ID card and passes it to the MQSeries Batch Interface transaction running under CICS. The user that starts the batch interface must be a surrogate for batch users who want to use the batch interface.

In this example, you use the MQADMIN user to start the batch interface and act as surrogate to any batch users (that is, STEVEJ). To register MQADMIN as a surrogate, you can issue the following command:

```
TSS ADD(MQADMIN) SURROGAT(STEVEJ)
```

Note: For the surrogate feature to be active, the facility matrix option XUSER=YES must be set.

When the batch user attempts to establish a connection to the queue manager, the batch interface user (MQADMIN) starts the partner transaction (MQBX) as the batch user. This is why the batch interface user must be a surrogate for the batch user.

From this point on, all MQSeries API calls issued by the batch user will be treated as if they were issued by the batch user under CICS. Therefore, the batch user should be granted the appropriate MQCONN and MQQUEUE privileges.

The batch user also needs authority to execute the MQBX transaction, for example:

```
TSS ADD(STEVEJ) TRANS(CICSPROD,MQBX)
```

Client user permissions

Client users should be treated the same as CICS application users. For security purposes, they are the same. MQSeries API calls issued by client programs are treated as if they were issued by the client user under CICS.

In this example, the client user SHELLYS has already been granted the necessary authority to get and put messages to the TEST.Q3 queue.

Java program clients are a special case for client user permissions. Existing MQSeries Java classes may attempt to open the queue manager as an object during an MQCONN request. This means, for such clients, the client user must have READ access to the queue manager object. For example:

```
TSS PER(SHELLYS) MQQUEUE(VSE.QM1.VSE.QM1) ACC(READ)
```

Command permissions

Authority to issue commands is required when command or command resource security is active. Command authority is required to create, modify, delete and display MQSeries objects such as the queue manager, channels and queues.

Commands can be issued via the MQSeries master terminal transaction (for example, MQMT), via PCF messages, and the MQSC command utility.

Command permissions involve resources belonging to the MQCMDS class. For a full list of these resources, and the permissions required for each command, refer to section “Resource definitions for command security” on page 296.

Full command authority can be granted to the MQADMIN user by issuing the following TSS command:

```
TSS PER(MQADMIN) MQCMDS(VSE.QM1) ACC(ALL)
```

Since all command resources are prefixed with the queue manager name (system identifier), the MQADMIN user will have ACC(ALL) to any and all of these resources.

Permissions for commands can also be granted by command type or for each individual command. For example, to grant permissions by command type to user JOHNS to issue DISPLAY commands, issue the following:

```
TSS PER(JOHNS) MQCMDS(VSE.QM1.DISPLAY) ACC(READ)
```

User JOHNS can now issue DISPLAY commands to examine the queue manager, channels and queues.Alternatively, to restrict JOHNS to DISPLAY commands for queues only, that is to restrict user JOHNS to individual commands rather than commands by type, issue the following:

```
TSS PER(JOHNS) MQCMDS(VSE.QM1.DISPLAY.QALIAS)  ACC(READ)
TSS PER(JOHNS) MQCMDS(VSE.QM1.DISPLAY.QLOCAL)  ACC(READ)
TSS PER(JOHNS) MQCMDS(VSE.QM1.DISPLAY.QREMOTE) ACC(READ)
```

User JOHNS can now issue DISPLAY commands for any type of queue, but not for the queue manager or channels.

Command resource permissions

If command resource security is active, command permissions are insufficient to issue commands against specific resources. A user must also be granted command resource permissions to those specific resources.

Command resource security should not be confused with command security. Command security restricts access to commands, whereas command resource security restricts issuing commands against specific resources. Consequently, command resource security is only relevant for commands that affect specific objects (that is, queues and channels).

Resources for command resource security are defined to the MQADMIN class.

Full command resource authority can be granted the MQADMIN user by issuing the following TSS command:

```
TSS PER(MQADMIN) MQADMIN(VSE.QM1) ACC(ALL)
```

Once again, since all command resources are prefixed with the queue manager name, the MQADMIN user will have ACC(ALL) to any and all of the command resources defined to the MQADMIN class.

Alternatively, since command resource types are limited to channels and queues, the MQADMIN user could be granted full command resource authority by issuing the following commands:

```
TSS PER(MQADMIN) MQADMIN(VSE.QM1.CHANNEL) ACC(ALL)
TSS PER(MQADMIN) MQADMIN(VSE.QM1.QUEUE)   ACC(ALL)
```

Like command security, permissions can be granted by object type or for each individual object. For example, to grant permissions by command resource type to user JOHNS to issue ALTER commands for any queue object, issue the following:

```
TSS PER(JOHNS) MQADMIN(VSE.QM1.QUEUE) ACC(ALTER)
```

User JOHNS can now issue ALTER commands to modify any queue providing he also has command authority to issue ALTER commands (assuming command security is active).

Alternatively, to restrict JOHNS to ALTER commands for queues TEST.Q1 and TEST.Q2, that is to restrict user JOHNS to individual objects rather than commands by type, issue the following:

```
TSS PER(JOHNS) MQADMIN(VSE.QM1.QUEUE.TEST.Q1) ACC(ALTER)
TSS PER(JOHNS) MQADMIN(VSE.QM1.QUEUE.TEST.Q2) ACC(ALTER)
```

User JOHNS can now issue ALTER commands for TEST.Q1 and TEST.Q2, but not for any other queue.

Command resource permissions are not required for DISPLAY commands.

Trigger permissions

Trigger programs and transactions are started automatically when an application program puts a message to a queue that is defined to start a trigger. The invocation and control of trigger instances is handled by MQSeries transaction MQ02.

Therefore, if an application user puts messages to a queue that may start a trigger instance, that user must have authority to run the MQ02 transaction.

In this example, you could define TEST.Q1 to start a trigger program every time a message is put to the queue. For example, the trigger program may get a message from TEST.Q1 and put a message on TEST.Q2. To enable application user JANED to put messages to TEST.Q1 and successfully start the trigger instance, you need to grant authority to the MQ02 transaction. For example:

```
TSS ADD(JANED) TRANS(CICSPROD,MQ02)
```

If you do not grant this authority, JANED can successfully put messages to the target queue, but the trigger instance will ABEND.

If you trigger a transaction, the application user must also have authority to run the trigger transaction. If you use program security, the application user needs authority to run the trigger program and a range of MQSeries programs (the exact programs are beyond the scope of this Appendix).

If you use the trigger option Allow Restart of Trigger in a queue definition, transaction MQSM will, when appropriate, attempt to run the MQ02 transaction. MQSM runs as the MQSeries startup user. Therefore, for security purposes, you should be careful when using the Allow Restart of Trigger feature. For details about this option, see “Trigger Information” on page 80.

CICS startup

Your CICS startup deck should include a // ID card. For the example user CICSP1, the // ID card would appear as follows:

```
// JOB jobname
// ID USER=CICSP1,PWD=P1CICS
```

You also need to identify the CICS default user as a SIT parameter. For the example user CICSP1DF, the SIT parameter would appear as follows:

```
DFLTUSER=CICSP1DF
```

Starting MQSeries

It is important that MQSeries is started by a user with sufficient authority. In this example, the user MQADMIN has full access to the MQSeries datasets, transactions, and MQSeries resources, including connection authority.

To start MQSeries, log on to CICS as MQADMIN and run the following transactions:

- MQSE
- MQIT

Another way to start MQSeries is to run 'MQSE I', or use the MQMT transaction, option 2.4.

A further option is to use the PLTPI program. The DFHPLT macro does not allow you to specify a userid with a PLTPI program. However, you can specify a SIT parameter for PLTPIUSR. For example:

```
PLTPIUSR=PLTUSER
```

In such a case, the PLTPIUSR must be authorized to the appropriate resources defined by PLTPISEC.

Remember that your PLTPIUSR may run programs that are not relevant to MQSeries for VSE/ESA, so, in this example implementation, it may not be appropriate to use user MQADMIN. Therefore, you can define a special PLTPIUSR as follows:

```
--  Create the PLTPIUSR
TSS CREATE(PLTUSER)  NAME('CICS PLTPI USER')  TYPE(USER)      +
                     DEPT(CICSP1G) PAS(PLTP1,0) FAC(CICSPROD)

--  Grant surrogate authority to CICS region user
TSS ADD(CICSP1) SURROGAT(PLTUSER)

--  Grant access to MQ File Control entries
TSS PER(PLTUSER) FCT(MQFCNFG) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFLOG)  ACC(ALL)
TSS PER(PLTUSER) FCT(MQFMON)  ACC(ALL)
TSS PER(PLTUSER) FCT(MQFERR)  ACC(ALL)
TSS PER(PLTUSER) FCT(MQFREOR) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFSSET) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFI001) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFI002) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFI003) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFO001) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFO002) ACC(ALL)
TSS PER(PLTUSER) FCT(MQFO003) ACC(ALL)

--  Grant authority to necessary transactions
TSS ADD(PLTUSER) TRANS(CICSPROD,INWL,IESO,IESN,MQIT,MQSM,MQTL)
```

Note that the CICS region user must be a surrogate for the PLTPIUSR. Once again, to activate the surrogate user feature, you should include the following facilities matrix option:

```
FACILITY(CICSPROD=XUSER=YES)
```

Also, take care that you do not grant NORESCHK to the PLTPIUSR. This is because resource checking for security switches will always result in success. Success indicates that the switch is present, and security features are deactivated. In other words, if the MQSeries startup user, PLTPI or otherwise, has NORESCHK authority, MQSeries resource security will be deactivated.

If you do not identify a SIT parameter for a PLTPIUSR, CICS TS uses the CICS default user. Although it may not be appropriate to authorize the CICS default user to MQSeries resources, it is possible to use the default user for MQSeries activation during CICS initialization.

Stopping MQSeries

The MQADMIN user has the authority to stop MQSeries by running the MQST transaction (because it has authority to all MQSeries transactions).

If you want to shut down MQSeries via the PLTSD, you must ensure that the shutdown user is authorized to the appropriate resources relevant to the PLTSD phase (these may be other than MQSeries resources).

The shutdown user is the user who issues the shutdown command, for example:

```
CEMT P SHUT
```

If this command is issued from the console, the console user must have authority similar to the PLTPIUSR user described earlier. Also, the shutdown user should have authority to execute the CEMT transaction.

Appendix I. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM documentation or non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those documents or Web sites. The materials for those documents or Web sites are not part of the materials for this IBM product and use of those documents or Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created

© Copyright IBM Corp. 1995, 2004

programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories, Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Copyright license

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

ACF/VTAM	AIX	AS/400
BookManager	CICS	CICS/VSE
IBM	Language Environment	MQSeries
MVS/ESA	OS/2	OS/390
OS/400	System/370	System/390
VSE/ESA	VTAM	

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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 MVS/ESA. A complete list of MQSeries for MVS/ESA abend reason codes and their explanations is contained in the *MQSeries for MVS/ESA Messages and Codes* manual.

active log. See *recovery log*.

adapter. An interface between MQSeries for MVS/ESA and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access MQSeries services.

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.

Adopt MCA. An MQSeries feature that allows an MCA instance to "adopt" the function of an existing MCA when it is deemed to have stalled.

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 MVS/ESA, a component of the CICS adapter that handles unscheduled events occurring as a result of connection requests to MQSeries for MVS/ESA.

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 MVS address space that is connected to MQSeries for MVS/ESA.

APAR. Authorized program analysis report.

application environment. The software facilities that are accessible by an application program. On the MVS platform, CICS and IMS are examples of application environments.

application queue. A queue used by an application.

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 open an MQSeries object.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current, unaltered release of a program.

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.

batch auto-start. A queue manager parameter used to indicate whether or not the MQSeries batch interface should be started automatically during system initialization.

batch identifier. An XPCC identifier used to uniquely identify a queue manager to batch applications.

© Copyright IBM Corp. 1995, 2004

BMS. Basic mapping support.

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.

buffer pool. An area of main storage used for MQSeries for MVS/ESA queues, messages, and object definitions. See also *page set*.

Bullet-proof. A feature of MQSeries channels that allows for a channel to wait no longer than a configurable period of time to receive data.

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.

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.

class. For security, a class associates a group of resources. MQSeries uses the security classes MQADMIN, MQCONN and MQQUEUE.

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*.

COA. Confirm-on-arrival. In reply queue processing, a reply message can be generated when a message is initially put to a queue by using the COA report option in the message descriptor of an object message.

COD. Confirm-on-delivery. In reply queue processing, a reply message can be generated when a message is initially read from a queue by using the COD report option in the message descriptor of an object message.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

command. In MQSeries, an instruction that can be carried out by the queue manager.

command processor. An MQSeries program responsible for processing PCF messages. The command processor validates and executes PCF commands, and generates response messages to the issuer.

command resource security. Security pertaining to MQSeries commands issued against MQSeries resources.

command server. An MQSeries program responsible for processing the system command queue. The command server reads PCF message from the command queue and starts an instance of the MQSeries command processor to process the PCF message.

command server auto-start. A queue manager parameter used to indicate whether or not theMQSeries command server should be started automatically during system initialization.

commit. An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *backout*.

completion code. A return code indicating how an MQI call has ended.

connect. To provide a queue manager connection handle, which an application uses on subsequent MQI calls. The connection is made either by the MQCONN call, or automatically by the MQOPEN call.

connection handle. The identifier or token by which a program accesses the queue manager to which it is connected.

context. Information about the origin of a message.

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.

CWS. CICS Web Support. A feature of CICS TS that allows CICS transactions to be run from a web browser.

D

DAE. Dump analysis and elimination.

datagram. The simplest message that MQSeries supports. This type of message does not require a reply.

DBCS. In data conversion, a Double Byte Character Set.

DCI. Data conversion interface.

DCT. In CICS, the Destination Control Table.

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.

deferred connection. A pending event that is activated when a CICS subsystem tries to connect to MQSeries for MVS/ESA before MQSeries for MVS/ESA has been started.

distributed application. In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

distributed queue management (DQM). In message queuing, the setup and control of message channels to queue managers on other systems.

DLQ. Dead-letter queue.

DQM. Distributed queue management.

dump analysis and elimination (DAE). An MVS service that enables an installation to suppress SVC dumps and ABEND SYSUDUMP dumps that are not needed because they duplicate previously written dumps.

E

environment. See *application environment*.

ESM. External security manager.

event. See *instrumentation 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 log. See *event queue*.

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.

exit. A program called at defined places in the processing carried out by the queue manager or MCA programs.

external security manager (ESM). A security product that is invoked by the MVS System Authorization Facility. RACF® is an example of an ESM.

F

FCT. In CICS, the File Control Table.

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 MVS/ESA, regardless of the state of any currently active tasks. Contrast with *quiesced shutdown*.

FRR. Functional recovery routine.

functional recovery routine (FRR). An MVS recovery/termination manager facility that enables a recovery routine to gain control in the event of a program interrupt.

G

get. In message queuing, to use the MQGET call to remove a message from a queue. See also *browse*.

H

handle. See *connection handle* and *object handle*.

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*.

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. In MQSeries, an event is a logical combination of conditions that is detected by a queue manager or channel instance.

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.

IP Address. Internet Protocol address. Usually a four-part dotted decimal value that uniquely identifies a remote host, for example, 1.20.33.444.

ISO. International Standards Organization. In data conversion, ISO code pages are those that conform to ISO definitions.

L

listener. A communications program that runs while MQSeries is active. The Listener program waits for connection requests from Sender MCAs or from client programs. For MQSeries for VSE/ESA V2.1, the Listener exclusively waits for TCP/IP connection requests and starts the Receiver MCA.

local definition. An MQSeries object belonging to a local queue manager.

local definition of a remote queue. An MQSeries object belonging to a local queue manager. This object defines the attributes of a queue that is owned by another queue manager. In addition, it is used for queue-manager aliasing and reply-to-queue aliasing.

local queue. A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

local queue manager. The queue manager to which a program is connected and that provides message queuing services to the program. Queue managers to which a program is not connected are called *remote queue managers*, even if they are running on the same system as the program.

log. In MQSeries, a file recording the work done by queue managers while they receive, transmit, and deliver messages.

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 non-recoverable.

MCA. Message channel agent.

MCI. Message channel interface.

message. In message queuing applications, a communication sent between programs. See also *persistent message* and *nonpersistent message*. In system programming, information intended for the terminal operator or system administrator.

message channel. In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender and a receiver) and a communication link. Contrast with *MQI channel*.

message channel agent (MCA). A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

message 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 exit. An exit program called during channel operation following the retrieval of a message from a queue and prior to a message being placed on a queue.

message expiry. Message attribute identifying a period of time expressed in tenths of a second. The message becomes eligible to be discarded if it has not been removed from the destination queue before this period of time elapses.

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*.

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*.

MQMD. MQSeries Message Descriptor. The MQMD is a data structure that is prefixed to all MQSeries messages.

MQSC. MQSeries Command. MQSC commands are verb-based text commands that manipulate or display MQSeries objects.

MQSeries. A family of IBM licensed programs that provides message queuing services.

MQSeries client. Part of an MQSeries product that can be installed on a system without installing the full queue manager. The MQSeries client accepts MQI calls from applications and communicates with a queue manager on a server system.

N

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 (MVS/ESA only), or a storage class (MVS/ESA 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.

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 MVS/ESA moves data (for example, queues and messages) from buffers in main storage to permanent backing storage (DASD).

PCF. Programmable Command Format. A data message containing an MQSeries command and associated parameters. PCF messages are written to the system command queue.

PCT. In CICS, the Program Control Table.

pending event. An unscheduled event that occurs as a result of a connect request from a CICS adapter.

performance event. A category of event indicating that a limit condition has occurred.

persistent message. A message that survives a restart of the queue manager.

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.

PKI. Public Key Infrastructure. The PKI infrastructure includes X.509 digital certificates used by SSL services.

platform. In MQSeries, the operating system under which a queue manager is running.

point of recovery. In MQSeries for MVS/ESA, the term used to describe a set of backup copies of MQSeries for MVS/ESA 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*.

port. A unique communications identifier used by TCP/IP programs to establish a conversation with a specific application. The target application binds a

TCP/IP socket to the unique port number and then waits for connection requests for the port from remote hosts.

PPT. In CICS, the Processing Program Table.

program temporary fix (PTF). A solution or by-pass of a problem diagnosed by IBM field engineering as the result of a defect in a current, unaltered release of a program.

PTF. Program temporary fix.

Q

queue. An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages—they point to other queues, or can be used as models for dynamic queues.

queue manager. 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.

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

reason code. A return code that describes the reason for the failure or partial success of an MQI call.

receive exit. An exit program called immediately following the receipt of data over an active channel.

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.

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.

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.

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.

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.

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.

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*.

RTM. Recovery termination manager.

S

SAF. System Authorization Facility. SAF is an interface between the VSE/ESA operating system and external security managers. The SAF interface is used for security purposes.

SBCS. In data conversion, a Single Byte Character Set.

security exit. An exit program called during the establishment of a channel.

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.

Send exit. An exit program called prior to the transmission of data over an active 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*.

session ID. In MQSeries for MVS/ESA, the CICS-unique identifier that defines the communication link to be used by a message channel agent when moving messages from a transmission queue to a link.

shutdown. See *immediate shutdown*, *preemptive shutdown*, and *quiesced shutdown*.

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*.

socket. A communications handle used by TCP/IP programs to send data to, and receive data from, a remote host.

SSID. Subsystem Identifier. An SSID is usually synonymous with an MQSeries queue manager name.

SSL. Secure Sockets Layer. A integrated feature of the TCP/IP product that provides a set of services to secure e-business transactions, including data encryptions and X.509 certificate exchange.

subsystem. In MVS, a group of modules that provides function that is dependent on MVS. For example, MQSeries for MVS/ESA is an MVS subsystem.

symptom string. Diagnostic information displayed in a structured format designed for searching the IBM software support database.

synchronous messaging. A method of communication between programs in which programs place messages on message queues. With synchronous messaging, the sending program waits for a reply to its message before resuming its own processing. Contrast with *asynchronous messaging*.

syncpoint. An intermediate or end point during processing of a transaction at which the transaction's protected resources are consistent. At a syncpoint, changes to the resources can safely be committed, or they can be backed out to the previous syncpoint.

system command queue. The system command queue is a communication parameter of the global system definition and identifies the target queue for PCF command messages.

system reply queue. The system reply queue is a communication parameter of the global system definition and identifies the target queue for MQSC response messages.

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 MVS/ESA target data set names.

task switching. The overlapping of I/O operations and processing between several tasks.

TCP/IP. Transmission Control Protocol, Internet Protocol. TCP/IP is a family of communications protocols.

termination notification. A pending event that is activated when a CICS subsystem successfully connects to MQSeries for MVS/ESA.

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*.

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.

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.

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

undelivered-message queue. See *dead-letter queue*.

Unicode. Codepage UCS-2 is the Universal Multiple-Octet Coded Character Set defined by ISO/IEC 10646-1:1993(EE).

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

X.509. X.509 is the standard used for the generation and interpretation of PKI certificates.

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 Tru64 UNIX, V5.1
- MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.2
- MQSeries for Compaq Tru64 UNIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for OS/390, V2.2
- MQSeries for SINIX and DC/OSx, V2.2
- MQSeries for Sun Solaris, 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

Any exceptions to this general rule are indicated.

MQSeries Brochure

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

MQSeries: An Introduction to Messaging and Queuing

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

MQSeries Intercommunication

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

MQSeries Queue Manager Clusters

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

This book applies to the following MQSeries products only:

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

MQSeries Clients

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

MQSeries System Administration

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

This book applies to the following MQSeries products only:

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

© Copyright IBM Corp. 1995, 2004

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

MQSeries Event Monitoring
MQSeries Event Monitoring, SC34-5760, describes how to use MQSeries instrumentation events.

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

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

This book applies to the following MQSeries products only:
- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

MQSeries Messages
The *MQSeries Messages* book, GC33-1876, which describes "AMQ" messages issued by MQSeries, applies to these MQSeries products only:
- MQSeries for AIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1

This book is available in softcopy only.

For other MQSeries platforms, the messages are supplied with the system. They do not appear in softcopy manual form.

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

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

MQSeries Programming Interfaces Reference Summary
The *MQSeries Programming Interfaces Reference Summary*, SX33-6095, summarizes programming interfaces information including the application programming interface, the application messaging interface, event messages, PCF messages, and installable services.

MQSeries Using C++
MQSeries Using C++, SC33-1877, provides both guidance and reference information for users of the MQSeries C++ programming-language binding to the MQI. MQSeries C++ is supported by these MQSeries products:
- MQSeries for AIX, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for OS/2 Warp, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for OS/390, V2.1
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1

MQSeries C++ is also supported by MQSeries clients supplied with these products and installed in the following environments:
- AIX
- HP-UX
- OS/2
- Sun Solaris
- Windows NT

- Windows 3.1
- Windows 95 and Windows 98

MQSeries Using Java

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

- MQSeries for AIX, V5.1
- MQSeries for AS/400, V5.1
- MQSeries for HP-UX, V5.1
- MQSeries for MVS/ESA V1.2
- MQSeries for OS/2 Warp, V5.1
- MQSeries for Sun Solaris, V5.1
- MQSeries for Windows NT, V5.1

This book is available in softcopy only.

MQSeries Application Messaging Interface

MQSeries Application Messaging Interface, SC34-5604, describes how to use the application messaging interface, which is an easy-to-use interface to the MQI.

MQSeries platform-specific publications

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

MQSeries for AIX

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

MQSeries for AS/400

MQSeries for AS/400 Version 5 Release 1 Quick Beginnings, GC34-5557

MQSeries for AS/400 Version 5 Release 1 System Administration, SC34-5558

MQSeries for AS/400 Version 5 Release 1 Application Programming Reference (ILE RPG), SC34-5559

MQSeries for AT&T GIS UNIX

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

MQSeries for Compaq (DIGITAL) OpenVMS

MQSeries for Compaq (DIGITAL) OpenVMS Version 2 Release 2.1.2 System Management Guide, GC33-1791

MQSeries for Compaq Tru64 UNIX

MQSeries for Compaq Tru64 UNIX, Version 5.1 Quick Beginnings, GC34-5684

MQSeries for HP-UX

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

MQSeries for OS/2 Warp

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

MQSeries for OS/390

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

MQSeries for OS/390 Version 2 Release 2 Program Directory

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

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

MQSeries for OS/390 Version 2 Release 2 Concepts and Planning Guide, SC34-5650

MQSeries for OS/390 Version 2 Release 2 System Setup Guide, SC34-5651

MQSeries for OS/390 Version 2 Release 2 System Administration Guide, SC34-5652

MQSeries Publish/Subscribe

MQSeries Publish Subscribe User's Guide, GC34-5269

MQSeries link for R/3

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

MQSeries for SINIX and DC/OSx

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

MQSeries for Sun Solaris

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

MQSeries for Tandem NonStop Kernel

MQSeries for Tandem NonStop Kernel Version 2 Release 2.0.1 System Management Guide, GC33-1893

MQSeries for VSE/ESA

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

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

MQSeries for Windows

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

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

MQSeries for Windows NT

MQSeries for Windows NT Version 5 Release 1 Quick Beginnings, GC34-5389

MQSeries for Windows NT Using the Component Object Model Interface, SC34-5387

MQSeries LotusScript Extension, SC34-5404

Softcopy books

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

BookManager format

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

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

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, V2.2
- 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 WebSphere MQ family Web site at:

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

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 WebSphere MQ family Web site is at:

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

By following links from this Web site you can:

- Obtain latest information about the WebSphere MQ family.
- Access the MQSeries books in PDF formats.
- Download MQSeries SupportPacs.

Index

A

access levels, queue security 292
activating SSL services 284
Adopt MCA
  check 57
  features 55
  parameters 56
alias queue
  altering parameters 263
  defining 266
  deleting definition 269
  displaying attributes 270
alias queue manager, creating 85
alias queues
  creating 84
  description 5
  security 293
alias reply, creating 86
allocating MQSeries files when installing 11
ALTER CHANNEL command 257
ALTER QALIAS command 263
ALTER QLOCAL command 264
ALTER QMGR command 274
ALTER QREMOTE command 266
altering local queue manager parameters 274
altering parameters of alias queue 263
altering parameters of local queue 264
altering parameters of remote queue 266
altering the parameters of a channel 257
APPC
  See LU 6.2
application
  data 1
  design considerations 155
  MQI administration support 357
  programming errors, examples of 149
  time-independent 1
applications
  C programming language 358
  design guidelines 358
  PL/I programming language 358
attributes of queues 4
authority checking
  PCF commands 174
auto-start
  batch interface 138
automatic reorganization 78

B

back out changes function call 321
batch connection security 291
batch interface
  auto-start 138
  data integrity 139
  how to use 138
  identifier 137
batch interface *(continued)*
  restrictions on use 140
  security 291, 504
  starting 138
  stopping 138
  using 136
  verifying 139
batch interface auto-start 72
batch interface identifier 71
batch modules 135
bibliography 521
BookManager 524
browse function 113
building applications 357
built-in format, data conversion 163
bullet-proof channels
  feature 57
  parameters 58

C

Change channel PCF command 176
Change Queue Manager PCF command 191
Change queue PCF command 183
changing
  MQXP TDQ definition 14
channel
  altering parameters 257
  configuration 456
    guidelines 36
  defining new 258
  definitions
    creating 88
    modifying 91
  deleting definition 260
  description 5
  displaying definition 260
  exit parameters 94
  monitoring 112
  opening and closing 104
  resetting message sequence number 262
  starting 263
  stopping 263
channel commands 256
channel events 75, 123
channel exit
  sample 55
channel exits
  See also exits
  See exits, channel
channel SSL parameters
  setting 92
channel-exit calls and data structures 51
channels
  bullet-proof
    See bullet-proof channels
checklist, security 296, 302
CICS
  connection definition 33
CICS *(continued)*
  file management 18
  initialization PLT (PLTPI) list 28
  installing table entries for MQSeries 16
  journal control table 18
  modifying start-up deck 17
  Program List Table Shut Down (PLTSD) 28
  recovery and restart 18
  security example 507
  session definition 34
  startup and shutdown 28
  startup JCL 306
  system definition 307
CICS web support 115
client
  configuration support 6
  connection security 291
  error messages on DOS and Windows 160
  Java program 292
  problem determination 159
  security example 505
  support for 465
close object function call 323
code page
  client conversion 465
  user defined
    creating 97
    deleting 98
    modifying 98
command line function 131
command messages
  issuing PCF 170
  PCF 170
command permissions 505
command resource permissions 506
command resource security 287
  resource definitions 300
command security 287
  MQMT options 299
  PCF messages 297
command security for MQSeries commands 298
command server
  PCF 169
command server auto-start 71
command server convert 71
command server DLQ store 71
commands
  See MQSeries commands (MQSC)
commit changes function call 324
communications process 117
compiling channel-exit programs 50
configuration functions
  channel definitions
    creating 88
    modifying 91
  code page definitions
    maintaining 96

© Copyright IBM Corp. 1995, 2004

configuration functions *(continued)*
user defined 97
display options
channel definitions 100
code page definitions 100
global system definition 99
queue definitions 100
global system definition 67
main menu 65
queue definitions, creating
alias queue 84
alias queue manager 85
alias reply 86
local queue 77
local queue extended definition 79
main screen 76
remote queue 83
queue definitions, modifying and deleting
deleting an existing queue 88
modifying an existing queue 87
selecting an existing queue 87
security 499
configuration guidelines
channel 36
example of 42
overview 35
queue 39
queue manager 35
configuration using MQMT 47
configuration using MQSC 50
configuration using PCF 49
configuration worksheet 451
configuring a channel for SSL 281
configuring network communications 31
configuring queue manager for SSL 280
confirm on arrival message, security 295
confirm on delivery message, security 295
connect queue manager function call 325
connection security 286
batch connection 291
client connection 291
resource definitions 290
console
messages 496
optional logging 72
conversion, data formats 163
built-in formats 163
exit program 164, 165
LE/VSE 164
Copy Channel PCF command 197
Copy Queue PCF command 200
CorrelId, performance considerations when using 156
Create Channel PCF command 202
Create Queue PCF command 204
creating
alias queue manager 85
alias queues 84
alias reply 86
channel definitions 88
local queue extended definition 79
local queues 77
queue manager 66
creating *(continued)*
remote queues 83
user defined code page 97
CWS
using with MQSeries 116
CWS converter program 116
CWS MQSeries modules 115

D

data conversion
built-in formats 163
exit program 164, 165
LE/VSE 164
messages 163
data formats, conversion 163
built-in formats 163
exit program 164, 165
LE/VSE 164
data responses to commands 226
data types, detailed description
MQDLHstructure 313
MQGMO structure 313
MQMD structure 314
MQOD structure 318
MQPMO structure 319
MQTM structure 320
dataset security 288
example 501
DCHFMT4 sample program 165
DCT
installation 17
sample entries 305
dead-letter header structure 313
dead-letter queue
description 5
security 294
specifying 67
debugging
common programming errors 149
preliminary checks 145
DEFINE CHANNEL command 258
DEFINE QALIAS command 266
DEFINE QLOCAL 267
DEFINE QREMOTE command 269
defining alias queue 266
defining installation configuration resources 22
defining local queue 267
defining new channel 258
defining queues 4
defining remote queue 269
delete all function 140
DELETE CHANNEL command 260
Delete Channel PCF command 205
DELETE QALIAS command 269
DELETE QLOCAL command 270
DELETE QREMOTE command 270
Delete Queue PCF command 206
deleting
channel definitions 91
queue definitions 88
user defined code page 98
deleting alias queue definition 269
deleting channel definition 260
deleting local queue definition 270
deleting remote queue definition 270
destination control table
See DCT
disabling events 126
disconnect queue manager function call 327
DISPLAY CHANNEL command 260
DISPLAY QALIAS command 270, 271, 273
DISPLAY QLOCAL command 271
DISPLAY QMGR command 276
DISPLAY QREMOTE command 273
displaying
channel definitions 100
code page definitions 100
global system definition 99
queue definitions 100
displaying alias queue attributes 270
displaying channel definition 260
displaying local queue attributes 271
displaying local queue manager parameters 276
displaying remote queue attributes 273
distributed queuing
dead-letter queue 5
incorrect output 158
undelivered-message queue 5
DOS clients error messages 160

E

enabling events 126
error codes
common to all PCF commands 175
error logs
viewing 130
Escape command response 227
Escape PCF command 208
event messages
See events, messages
event queue 5
event queues 128
Event queues 74
event-driven processing 1
events
channel 75, 123
disabling 126
enabling 126
messages 129
format 129
performance 75, 123
Qmgr 74
queue manager 121
queues 128
service interval 125
examples, programming errors 149
exit programs in CICS 51
exits
channel 43
configuring 47
message 46
receive 44
security 43
send 44
configuring
channel 47
conversion 164
sample 165

exits *(continued)*
data structures 51
entry point 52
writing 51
external security manager 285
configuration 500

F

FCT
event entries 305
PCF entries 304
sample entries 303
file control table
See FCT
function calls, detailed description
MQBACK 321
MQCLOSE 323
MQCMIT 324
MQCONN 325
MQDISC 327
MQGET 328
MQINQ 332
MQOPEN 339
MQPUT 343
MQPUT1 346
MQSET 350
function key operations 65

G

get message options structure 313
get queue manager function call 328
global system definition
entry fields 67
guidelines
backing up configuration file 75
queue manager name 66
specifying a dead-letter queue 67
glossary 513
guidelines, product configuration 35

H

HTML source file 115

I

identifier
batch interface 137
incorrect output 156
initializing MQSeries for VSE/ESA 19
initializing system 106
Inquire Channel command response 227
Inquire Channel Names command response 230
Inquire Channel Names PCF command 212
Inquire Channel PCF command 209
inquire object attributes function call 332
Inquire Queue command response 231
Inquire Queue Manager command response 235
Inquire Queue Manager PCF command 217
Inquire Queue Names command response 239
Inquire Queue Names PCF command 220
Inquire Queue PCF command 214
installation
allocating MQSeries files 11
CICS startup and shutdown 28
defining local queues 21
defining system log 20
defining SYSTEM.LOG queue 22
initializing the system 19
local queue verification test 23
migration procedure, steps required 28
modifying CICS start-up deck 17
preparing CICS table entries for MQSeries 16
procedures for new users 11
product 9
restoring MQSeries library 10
samples 7
security 12
specify the queue manager name 19
starting MQSeries for VSE/ESA 18
tape contents 7
target installation library 9
uppercase translation 18
verification test 23
VSAM installation catalog 9
installing SSL 279
instrumentation events 120
intercommunication example 451, 465
issuing
PCF command messages 170
issuing MQSeries commands 254
issuing MQSeries commands from batch 254

J

Java program clients
security 292
security example 505

K

key-ring member 70
key-ring sublibrary 70

L

LE/VSE, data conversion 164
library tape 7
licensed clients 280
listener task
starting 278
local queue
altering parameters 264
altering queue manager parameters 274
defining 267
deleting definition 270
displaying attributes 271
local queue *(continued)*
displaying queue manager parameters 276
testing queue manager responsiveness 278
local queue extended definition, creating 79
local queues
creating 77
dead-letter 5
defining during installation 21
description 4
transmission 5
undelivered-message 5
log, system 159
LU 6.2
connections 31, 33, 451
sessions 34

M

main configuration menu 65
maintaining
code page definitions 96
maintenance, queue 107
master terminal 63
displays 61
transactions 64
Max TCP/IP Wait 39
message
containing unexpected information 158
data conversion 163
description 1
descriptor 1
errors on DOS and Windows clients 160
information 468
lengths of 2
not appearing on queues 157
performance affected by length 156
queuing 1
retrieval algorithms 2
searching for particular 156
undelivered 159
variable length 156
message channel agent 117
Message channel agent
Adopt MCA 55
message data conversion 163
built-in formats 163
exit program 164, 165
LE/VSE 164
message descriptor structure 314
message expiry 118
message queue interface
See MQI
message queuing 1
message security 286
message sequence number
resetting for channel 262
message sequence number, resetting 105
message-driven processing 1
messages
descriptor 171
migrating from an earlier release 28

modifying
channel definitions 91
queue definitions 87
user defined code page 98
monitoring functions
channels 112
main menu 109
queues 110
MQ_CHANNEL_EXIT 52
MQBACK call 321
MQCD - channel data structure 55
MQCFIL structure 247
MQCFIN structure 243
MQCFSL structure 248
MQCFST structure 244
MQCL 131
MQCLOSE call 323
MQCMIT call 324
MQCONN call 325
MQCXP - channel exit parameter structure 55
MQDATA 160
MQDISC call 327
MQDLH structure 313
MQGET call 328
security issues 293
MQGMO structure 313
MQGMO-CONVERT option 163
MQI
description 1
local administration support 357
monitor 151
queue manager calls 4
sample programs 365
MQINQ call 332
MQIT transaction 19
MQJCONFG.Z 11
MQJQUEUE.Z 11
MQJSETUP.Z 11
MQMD structure 314
MQMT 63, 287
MQMT options
command security 299
MQOD structure 318
MQOPEN call 339
MQPCMD utility program 132
MQPMO structure 319
MQPMQSC program 254
MQPMQSC sample JCL 255
MQPMQSC utility program 254
MQPREORG function 141
MQPUT and MQPUT1, performance considerations 156
MQPUT call 343
security issues 293
MQPUT1 call 346
MQPUTIL 135
MQSC
See MQSeries commands (MQSC)
MQSC utility program 254
MQSE transaction 19
MQSeries commands
channel 256
command security 298
command structure 253
individually described 256
issuing 254
MQSeries commands *(continued)*
prerequisites before batch processing 256
queue commands 263
queue manager commands 273
rules for use 253
special characters 254
MQSeries commands (MQSC)
purpose 253
MQSeries for VSE/ESA
allocating files when installing 11
building applications 357
channel configuration 456
channel definition 36
client support 465
command line function 131
configuration 456
configuration worksheet 451
definition in CICS 31
LU 6.2 connection 451
overview 8
running applications 357
MQSeries publications 521
MQSET call 350
MQSU transaction 19
MQTM structure 320
MQXP TDQ definition
changing 14
MsgId, performance considerations when using 156

N

names of objects 3
network configuration 31
new users, procedures for 11

O

object descriptor structure 318
objects
administering using web browser 114
names 3
queue 4
queue manager in MQI calls 4
types 3
open object function call 339
opening and closing channels 104
operations functions
open and close channels 104
queue maintenance 107
reset message sequence number 105
start and stop queues 101
system initialization 106
operator function keys 65
optional logging 73
optional tracing
See tracing
overview of MQSeries for VSE/ESA 8

P

panel layout 62
PCF
command messages 170
PCF *(continued)*
command server
activating 169
stopping 169
commands
authority checking 174
common error codes 175
data responses 226
individual definitions 174
issuing 170
message descriptor 171
responses 172
security 297
structure 240
description 167
introduced 167
messages
command security 297
header 240
parameters 71
preparation 168
system command queue 168
using 170
PDF (Portable Document Format) 524
performance considerations
advantages of MQPUT1 156
application design 155
CorrelId 156
message length 156
MsgId 156
variable message length 156
when using trace 159
performance events 75, 123
permanent queues 2
PING QMGR command 278
Ping Queue Manager PCF command 221
PLTPI 28
PLTSD 28
Portable Document Format (PDF) 524
predefined queues 2
preparing
MQSeries for PCF 168
prerequisites for batch processing
MQSeries commands 256
problem determination
clients 159
incorrect output
messages containing unexpected information 158
messages not appearing on queues 157
with distributed queuing 158
network operation
SNA problems 146
TCP/IP problems 147
programming errors 149
trace 159
with local queue operation 145
procedures (installation) for new users 11
processing, event-driven 1
product configuration 35
Programmable Command Formats
See PCF
programmable system management 167
programming errors, examples of 149

publications
MQSeries 521
publications, MQSeries 521
put message function call 343
put message options structure 319
put one message function call 346

Q

Qmgr events 74
queue
alias 5
altering parameters 263
defining 266
deleting definition 269
displaying attributes 270
attributes of 4
configuration 39
dead-letter, specifying 67
defining 4
description 2
distributed, incorrect output from 158
event 5
event attributes 81
for MQSeries applications 357
inbound status 111
local
altering parameters 264
altering queue manager parameters 274
defining 267
deleting definition 270
displaying attributes 271
displaying queue manager parameters 276
testing queue manager responsiveness 278
monitoring 110
objects
alias 5
local 4
remote 4
outbound status 111
predefined 2
remote 4
altering parameters 266
defining 269
deleting definition 270
displaying attributes 273
starting and stopping 101
transmission 5
trigger attributes 80
queue commands 263
queue definition
deleting 88
main screen 76
modifying 87
selecting 87
queue maintenance 107
queue management
simplifying 167
queue manager
communications settings 69
configuration file, backing up 75
creating 66
description 4
queue manager *(continued)*
event settings 74
log settings 72
maximums 68
messages 467
name, specifying during installation 19
object in MQI calls 4
SSL parameters 281
status 102
TCP/IP settings 280
trace settings 72
unique name 66
queue manager commands 273
queue manager events 121
queue security 286
access levels 292
alias queues 293
resource definitions 292

R

recovery 18
remote administration
using PCFs 170
remote queue
altering parameters 266
defining 269
deleting definition 270
displaying attributes 273
remote queues
creating 83
description 4
security 294
reorganization, automatic 78
reorganizing queue files 142
reply queue security 295
requirements
hardware 8
software 8
RESET CHANNEL command 262
Reset Channel PCF command 222
resetting message sequence number 105
resetting message sequence number for a channel 262
resource definitions
command resource security 300
resources
security 286, 288
security example 503
switch 289
responses
to PCF commands 172
restoring installation library 10
retrieval algorithms for messages 2
return codes 149
running applications 357

S

sample code 358
sample JCL 142
MQPMQSC 255
sample programs 7
sample security configuration 499
Secure Sockets Layer services
See SSL services
security 285
batch connection 291
batch interface, example 504
checklist 302
classes 288
client connection 291
command 287
command resource 287
connection 286
resource definitions 290
dataset 288
dataset example 501
dead-letter queues 294
example implementation 499
external security manager 500
installing 12
message 286
queue 286
alias queues 293
remote queues 294
reply queues 295
resource definitions 292
system queues 295
resource definitions 296
resources 286, 288
example 503
switch resources 289
starting MQSeries 507
stopping MQSeries 508
transaction example 502
trigger program example 507
selecting queue definitions 87
selecting transmission protocol 89
sending
user data 172
set object attributes function call 350
simplifying
queue management 167
softcopy books 524
special characters
MQSC 254
specified operating environment 8
SSL
channels
parameters 282
problems with cipher specification support 161
problems with client authentication failure 162
problems with enabled 160
problems with general channel failure 162
problems with SSL availability 161
cipher specification 93, 282
client authentication 93, 283
configuring a channel 281
installing 279
parameters 70
setting channel 92
peer attributes 94, 283
problems
investigating 148
queue manager
configuring 280

SSL *(continued)*
services
activating 284
SSL feature
See SSL
SSL services 279
START CHANNEL command 263
Start Channel Listener PCF command 224
Start Channel PCF command 223
START LISTENER command 278
starting a channel 263
starting a TCP/IP listener task 278
starting and stopping queues 101
starting CICS 28
security 507
starting MQSeries 18
example 507
starting the batch interface 138
STOP CHANNEL command 263
Stop Channel PCF command 225
stopping a channel 263
stopping the batch interface 138
structure
PCF commands and responses 240
syncpoints and triggers 359
system command queue 71, 168
system initialization 106
system log 159
defining during installation 20
queue, defining 22
system messages 467
system operation 61
system queue security 295
System reply queue 71
system wait interval 35

T

tape contents 7
target installation library 9
TCP/IP
defining a receiver channel 463
defining a sender channel 463
establishing a connection 456
installation requirements 31
selecting 89
terminology used in this book 513
testing local queue manager responsiveness 278
time-independent applications 1
tracing 73
performance considerations 159
transactional interface 131
transient data queues
MQER 13
MQIE 15
MQXP 14
transmission protocol, selecting 89
transmission queue 5
trigger message structure 320
triggering 361
types of objects 3

U

unauthorized access, protecting from 285
undelivered message queue
See dead-letter queue
unit of work 359
uppercase translation 18
user authority for EXPIRY 296
user data
sending 172
using the batch interface 136
utility program
MQSC 254

V

verification, installation 23
VSAM file maintenance
automatic reorganization 78
creating space 140
delete all function 140
MQPREORG function
description of 141
sample JCL 142
reorganizing queue files 142
VSAM installation catalog 9
VSE/ESA system requirements 8

W

web browser
administering objects 114
web support
CICS 115
Windows clients error messages 160
Windows Help 524
writing channel-exit programs 50

Sending your comments to IBM

MQSeries® for VSE/ESA

System Management Guide

GC34-5364-04

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:
  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER
  Hampshire
  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 you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

© Copyright IBM Corp. 1995, 2004

Printed in USA

GC34-5364-04

Spine information:

MQSeries® for VSE/ESA

MQSeries for VSE/ESA System Management Guide

Version 2
Release 1 Modification 2