

MAINVIEW[®] AutoOPERATOR for MQSeries Installation and User Guide

Version 6.3.01

November 21, 2003



Copyright 2003 BMC Software, Inc. All rights reserved.

BMC Software, the BMC Software logos, and all other BMC Software product or service names are registered trademarks or trademarks of BMC Software, Inc. IBM and DB2 are registered trademarks of International Business Machines Corp. All other trademarks belong to their respective companies.

BMC Software considers information included in this documentation to be proprietary and confidential. Your use of this information is subject to the terms and conditions of the applicable End User License Agreement for the product and the proprietary and restricted rights notices included in this documentation.

Restricted Rights Legend

U.S. Government Restricted Rights to Computer Software. UNPUBLISHED -- RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure of any data and computer software by the U.S. Government is subject to restrictions, as applicable, set forth in FAR Section 52.227-14, DFARS 252.227-7013, DFARS 252.227-7014, DFARS 252.227-7015, and DFARS 252.227-7025, as amended from time to time. Contractor/Manufacturer is BMC Software, Inc., 2101 CityWest Blvd., Houston, TX 77042-2827, USA. Any contract notices should be sent to this address.

Contacting BMC Software

You can access the BMC Software Web site at <http://www.bmc.com>. From this Web site, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address BMC Software, Inc.
2101 CityWest Blvd.
Houston TX 77042-2827

Telephone 713 918 8800 or
800 841 2031

Fax 713 918 8000

Outside United States and Canada

Telephone (01) 713 918 8800

Fax (01) 713 918 8000

Customer Support

You can obtain technical support by using the Support page on the BMC Software Web site or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, please see “Before Contacting BMC Software.”

Support Web Site

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at <http://www.bmc.com/support.html>. From this Web site, you can

- read overviews about support services and programs that BMC Software offers
- find the most current information about BMC Software products
- search a database for problems similar to yours and possible solutions
- order or download product documentation
- report a problem or ask a question
- subscribe to receive e-mail notices when new product versions are released
- find worldwide BMC Software support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by Telephone or E-mail

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813. Outside the United States and Canada, please contact your local support center for assistance. To find telephone and e-mail contact information for the BMC Software support center that services your location, refer to the Contact Customer Support section of the Support page on the BMC Software Web site at www.bmc.com/support.html.

Before Contacting BMC Software

Before you contact BMC Software, have the following information available so that Customer Support can begin working on your problem immediately:

- product information
 - product name
 - product version (release number)
 - license number and password (trial or permanent)
- operating system and environment information
 - machine type
 - operating system type, version, and service pack or other maintenance level such as PUT or PTF
 - system hardware configuration
 - serial numbers
 - related software (database, application, and communication) including type, version, and service pack or maintenance level
- sequence of events leading to the problem
- commands and options that you used
- messages received (and the time and date that you received them)
 - product error messages
 - messages from the operating system, such as `file system full`
 - messages from related software



Contents

About This Book	xv
------------------------------	-----------

Chapter 1	What MAINVIEW AutoOPERATOR for MQSeries Is
	Understanding MAINVIEW AutoOPERATOR for MQSeries 1-1
	Understanding Automation for MQSeries Events 1-2
	Understanding Terms and Concepts 1-2
	What Messaging and Queuing Is 1-2
	What a Queue Manager Is 1-3
	How Queue Managers Communicate. 1-4
	What MQSeries Events Are 1-5
	Interactions between MAINVIEW AutoOPERATOR and MQSeries ... 1-8

Chapter 2	Implementing MAINVIEW AutoOPERATOR for MQSeries
	How AutoOPERATOR Connects to MQSeries 2-1
	Required Software 2-3
	Installation Overview 2-4
	Step 1. Activating MAINVIEW AutoOPERATOR for MQSeries. 2-6
	Specifying Product Option Password Keys 2-6
	Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card 2-8
	Modifying AAOPRM00 Parameter Settings 2-8
	Modifying AAOMQL00 Parameter Settings (Optional) 2-9
	Enabling Distributed Diagnostic Rules 2-11
	MAINVIEW AutoOPERATOR for MQSeries Security Considerations. 2-12
	Restarting the BBI-SS PAS 2-13
	Define Queue for Non-OS/390 Instrumentation Events 2-13
	Step 2. Enabling Instrumentation Events 2-13
	Enabling Instrumentation Events for Non-OS/390 Queue Managers. 2-14
	Enabling Instrumentation Events for OS/390 Queue Managers ... 2-14

Step 3. Defining Connectivity	2-14
Making Queue Manager Connections Using Command MQ On Ramp	2-15
Hyperlinking to Command MQ On Ramp in Your Web Browser . . .	2-16
Accessing Command MQ On Ramp Interface	2-17
Specifying the OS/390 Queue Manager	2-18
Specifying Non-OS/390 Queue Managers	2-18
Completing the Queue Manager Connections	2-20
Logging Off	2-21
Diagnostics	2-22
Connecting Queue Managers Manually	2-23
Step 4. Using the Installation Verification Procedure	2-27
Step 5. Setting Up MAINVIEW AutoOPERATOR for MQSeries to Co-Exist with PATROL for WebSphere MQ	2-31
MAINVIEW AutoOPERATOR for MQSeries BBI Commands	2-33

Chapter 3

Customizing MAINVIEW AutoOPERATOR for MQSeries

Parameters for BBPARM Member AAOPRMxx	3-1
Generating Instrumentation Events	3-3
Encountering Queues Set to Get(Disabled)	3-3
Encountering Queues Set as NOSHARE	3-4
Parameters for BBPARM Member AAOMQLxx	3-5
Specifying Queues	3-5
Example of Specifying Queues in AAOMQLxx	3-7

Chapter 4

Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics

Using the MAINVIEW AutoOPERATOR for MQSeries Workstation . .	4-1
MQSeries Workstation Fields	4-5
Performance Statistics.	4-5
Viewing Queue Management	4-7

Chapter 5

Automating MQSeries Events

How MAINVIEW AutoOPERATOR for MQSeries Automates MQSeries Events	5-1
What a Rule Is.	5-2
What MQS Events Are	5-2
Rule Processor MQSeries Variables	5-4
Enabling MQS Events	5-5
Creating Rules for Event Messages and Non-Event Messages.	5-7
Creating a Rule for a MQFMT_EVENT Message	5-7
Creating a Rule for a Non-MQFMT_EVENT Message.	5-21
MQSeries Instrumentation Events.	5-32
Variables for MQSeries Events	5-55
SHARED Variables.	5-55
Variables Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages	5-55

MQS Selection Criteria and Action Specification Panel Fields	5-63
Selection Criteria for MQS Panel.	5-63
Action Specification for MQS Panel	5-67
Tracking Automation Statistics for MQS Events.	5-71

Chapter 6

Using MAINVIEW AutoOPERATOR for MQSeries Solutions

Implementation Considerations	6-1
System Queue Archival	6-2
Using the Dead Letter Queue Solution.	6-2
Security Considerations	6-3
Implementing the Dead Letter Queue Solution	6-3
Stopping the Dead Letter Queue Solution	6-7
Using the System Queue Archival Solution	6-8
Security Considerations	6-8
Implementing the System Queue Archival Solution	6-9
Enabling the System Queue Archival Rules	6-10
Stopping the System Queue Archival Solution	6-14
Using the Service Interval Performance Solution.	6-15
Implementing the Service Interval Performance Solution	6-15
Stopping the Service Interval Performance Solution.	6-17
Using the Queue Depth Management Solution	6-17
Implementing the Queue Depth Management Solution.	6-18
Stopping the Queue Depth Management Solution.	6-19
Using the Channel Availability Solution	6-20
Implementing the Channel Availability Solution.	6-21
Stopping the Channel Availability Solution.	6-23
Using the Basic Intercommunication Solution	6-24
Security Considerations	6-24
Before You Begin.	6-24
Invoking the Basic Intercommunication Solution	6-27
Enabling the Basic Intercommunication Solution	6-27
Stopping the Basic Intercommunication Solution	6-28

Chapter 7

Command MQ Automation Power Line (APL)

Variables Returned from APL.	7-4
Examples.	7-6
Example 1 – Displaying Attributes of Queue Manager	7-6
Example 2 – Requesting Names of Queue Managers	7-6
Example 3 – Start Queue Manager	7-7
Example 4 – Stop Queue Manager.	7-7
Example 5 – Passing Multiple Commands	7-7

Chapter 8

Securing MAINVIEW AutoOPERATOR for MQSeries

Chapter 9**Using MAINVIEW AutoOPERATOR for MQSeries IMFEXEC Statements**

MQI Commands	9-2
IMFEXEC MQI Variables	9-2
What the MAINVIEW AutoOPERATOR MQI EXEC Interface Is	9-4
How Applications Communicate with MQSeries Objects	9-4
How Applications Interact with MQSeries Objects	9-4
How Completion and Reason Codes Are Returned	9-5
Creating Messages from Variables	9-6
Message Descriptor	9-10
Message Buffer	9-10
Dead Letter Header	9-10
DLH	9-10
Message Buffer	9-12
MQI BACK	9-13
MQI CLOSE	9-15
MQI CMIT	9-17
MQI CONN	9-19
MQI DISC	9-21
MQI GET	9-23
MQI OPEN	9-33
MQI PUT	9-37
MQI PUT1	9-47
CMD (Issue IMFEXEC Command to MQSeries with Response)	9-58
COPY MQI	9-63
DISPLAY MQI	9-66

Appendix A**Diagnosing MAINVIEW AutoOPERATOR for MQSeries Errors**

Where You Can Find Additional Information	A-1
Which Tools Are Available to Diagnose MAINVIEW AutoOPERATOR for MQSeries Problems	A-2
Diagnostic Checklist	A-4
Frequently Asked Questions	A-6
Examples of Specific Automation Problems and Resolutions	A-19
Dead Letter Queue Solution	A-19
Non-OS/390 Command Times Out	A-24
Rule Will Not Enable a Queue	A-27

Appendix B**MQI Sample Coding**

Appendix C

Rules Variables

EVENT Variables – Rules and Rule-Invoked EXECs	C-1
Dead Letter Header Variables – Rules and Rule-Invoked EXECs	C-3
Trigger Message Variables – Rules and Rule-Invoked EXECs	C-5
CICS Information Header Variables – Rules and Rule-Invoked EXECs	C-6
IMS Information Header Variables – Rules and Rule-Invoked EXECs	C-10
Transmission Queue Header Variables – Rules and Rule-Invoked EXECs	C-12
Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs	C-17
Work Information Header Variables – Rules and Rule-Invoked EXECs	C-19
Reference Message Header Variables – Rules and Rule-Invoked EXECs	C-20
Trigger Message 2 (Character Format) Variables – Rules and Rule-Invoked EXECs	C-22
Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs	C-23
Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs	C-24
PCF Variables – Rules and Rule-Invoked EXECs	C-25

Appendix D

EXECs Variables

General Purpose Variables – EXECs	D-1
Options and Miscellaneous Variables	D-2
EVENT Variables – EXECs	D-5
Dead Letter Header Variables – EXECs	D-8
Trigger Message Variables – EXECs	D-10
CICS Information Header Variables – EXECs	D-11
IMS Information Header Variables – EXECs	D-15
Transmission Queue Header Variables – EXECs	D-17
Message Descriptor Extension Variables – EXECs	D-22
Work Information Header Variables – EXECs	D-24
Reference Message Header Variables – EXECs	D-25
Trigger Message 2 (Character Format) Variables – EXECs	D-27
Rules and Formatting Header Variables – EXECs	D-28
Rules and Formatting Header Variables Version 2 – EXECs	D-29
PCF Variables – EXECs	D-30

Appendix E

Conversion Checklist

Converting to the Current Release of MAINVIEW AutoOPERATOR	E-1
Rule Processing	E-1
EXEC Manager	E-3

Index

Figures

Figure 1-1	Example of Channels	1-5
Figure 1-2	Interaction between MAINVIEW AutoOPERATOR and an OS/390 Queue Manager	1-9
Figure 1-3	Relationship between MAINVIEW AutoOPERATOR and an OS/390 and Non-OS/390 Queue Manager	1-10
Figure 2-1	Rule Processor Variable Dependencies Panel	2-31
Figure 3-1	Example of Coding in AAOMQLxx Member	3-7
Figure 4-1	MQSeries Workstation Panel	4-2
Figure 4-2	MQSeries Workstation Panel Fields	4-5
Figure 4-3	MQSeries Workstation - Performance Statistics	4-5
Figure 4-4	Queue Management Portion of the MQSeries Workstation Panel	4-7
Figure 5-1	Creating an MQFMT_EVENT Rule: Automation Control Panel (Example 1)	5-8
Figure 5-2	Creating an MQFMT_EVENT Rule: Rule Set Overview Panel (Example1)	5-10
Figure 5-3	Creating a MQFMT_EVENT Rule: Rule Processor Detail Control Panel (Example 1)	5-10
Figure 5-4	Creating a MQFMT_EVENT Rule: Rule Processor Detail Control Panel (Example 2)	5-11
Figure 5-5	Creating a MQFMT_EVENT Rule: Selection Criteria - MQS Panel (Example 1)	5-12
Figure 5-6	Selection Criteria - MQS Panel (Example 2)	5-13
Figure 5-7	Example of a MQSeries Event Types Panel	5-13
Figure 5-8	Example of a Help Panel for Q_DEPTH_HIGH	5-14
Figure 5-9	Selection Criteria - MQS Panel (Example 3)	5-15
Figure 5-10	Variable Dependencies - MQS Panel (Example 1)	5-16
Figure 5-11	Creating a MQFMT_EVENT Rule: Action Specification - MQS Panel (Example 1)	5-16
Figure 5-12	Action Specification - MQS Panel (Example 2)	5-17
Figure 5-13	Rule Processor Detail ControPanel (Example 3)	5-19
Figure 5-14	Rule Set Overview Panel (Example 2)	5-19
Figure 5-15	Confirming Rule Set Modifications Panel (Example 1)	5-20
Figure 5-16	Creating a Rule for a Message with Format MQFMT_STRING: Automation Control Panel (Example 1)	5-22

Figure 5-17	Creating a Rule for a Message with Format MQFMT_STRING: Automation Control Panel (Example 2)	5-23
Figure 5-18	Creating a Rule for a Message with Format MQFMT_STRING: Rule Set Overview Panel (Example 1)	5-23
Figure 5-19	Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 1)	5-24
Figure 5-20	Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 2)	5-25
Figure 5-21	Creating a Rule for a Message with Format MQFMT_STRING: Selection Criteria - MQS Panel (Example 1)	5-25
Figure 5-22	Selection Criteria - MQS Panel (Example 2)	5-26
Figure 5-23	Creating a Rule for a Message with Format MQFMT_STRING: Action Specification Panel	5-27
Figure 5-24	Action Specification - MQS Panel (Example 2)	5-28
Figure 5-25	Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 3)	5-30
Figure 5-26	Rule Set Overview Panel (Example 2)	5-30
Figure 5-27	Confirming Rule Set Modifications Panel (Example 2)	5-31
Figure 5-28	Selection Criteria - MQS Field Description	5-63
Figure 5-29	Action Specification - MQS Field Description	5-68
Figure 5-30	Example of EVNT Automation Reporter Record	5-71
Figure 5-31	Example of ACTN Automation Reporter Record	5-72
Figure 7-1	Example of Information Returned in Variables APLNOL and APLLN1 through APLLN13	7-5
Figure 9-1	Simple Message	9-7
Figure 9-2	Complex message	9-8
Figure 9-3	Existing Message	9-9
Figure 9-4	Updating Message Buffer and Dead Letter Header	9-10
Figure 9-5	Creating a Message Descriptor, Message Buffer and an IMS Information Header	9-12

Tables

Table 2-1	Installation Activities — Summary	2-4
Table 3-1	BBPARM Member AAOPRMxx Parameters	3-2
Table 3-2	AAOMQLxx Parameters	3-5
Table 4-1	MQSeries Workstation Panel Primary Commands	4-3
Table 4-2	Performance Statistics Field Names	4-6
Table 4-3	Queue Management Field Names	4-7
Table 5-1	Message Format and Description	5-3
Table 5-2	MQSeries Instrumentation Events	5-32
Table 5-3	SHARED Variables for MAINVIEW AutoOPERATOR for MQSeries	5-55
Table 5-4	Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages	5-56
Table 5-5	Selection Criteria - MQS Panel's Field Names	5-64
Table 5-6	Action Specification - MQS Panel's Field Names	5-68
Table 6-1	Rules from the Rule Set AAORULBR	6-25
Table 7-1	APL parameters	7-2
Table 7-2	APL Variables IMFRC, APLCC, and APLRC	7-4
Table 9-1	IMFEXEC Command Statements	9-2
Table 9-2	MQI EXEC Completion Codes	9-5
Table 9-3	Common Input and Output Variables for the IMFEXEC MQI PUT Statement	9-40
Table 9-4	Variables and Source	9-49
Table B-1	MQI Code Examples	B-1
Table C-1	EVENT Variables – Rules and Rule-Invoked EXECs	C-1
Table C-2	Dead Letter Header Variables – Rules and Rule-Invoked EXECs	C-3
Table C-3	Trigger Message Variables – Rules and Rule-Invoked EXECs	C-5
Table C-4	CICS Information Header Variables – Rules and Rule-Invoked EXECs	C-6
Table C-5	IMS Information Header Variables – Rules and Rule-Invoked EXECs	C-10
Table C-6	Transmission Queue Header Variables – Rules and Rule-Invoked EXECs	C-12
Table C-7	Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs	C-17

Table C-8	Work Information Header Variables – Rules and Rule-Invoked EXECs	C-19
Table C-9	Reference Message Header Variables – Rules and Rule-Invoked EXECs	C-20
Table C-10	Trigger Message 2 (Character format) Variables – Rules and Rule-Invoked EXECs	C-22
Table C-11	Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs	C-23
Table C-12	Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs	C-24
Table C-13	PCF Variables – Rules and Rule-Invoked EXECs	C-25
Table D-1	Event Variables - EXECs	D-5
Table D-2	Dead Letter Header Variables - EXECs	D-8
Table D-3	Trigger Message Variables - EXECs	D-10
Table D-4	CICS Information Header Variables - EXECs	D-11
Table D-5	IMS Information Header Variables - EXECs	D-15
Table D-6	Transmission Queue Header Variables - EXECs	D-17
Table D-7	Message Descriptor Extension Variables - EXECs	D-22
Table D-8	Work Information Header Variables - EXECs	D-24
Table D-9	Reference Message Header Variables - EXECs	D-25
Table D-10	Trigger Message 2 (Character format) Variables – EXECs	D-27
Table D-11	Rules and Formatting Header Variables – EXECs	D-28
Table D-12	Rules and Formatting Header Variables Version 2 – EXECs	D-29
Table D-13	PCF Variables – (Rules and Rule-Invoked EXECs)	D-30

About This Book

This book contains detailed information about MAINVIEW AutoOPERATOR for MQSeries and is intended for operators and system programmers, or anyone who plans to use this MAINVIEW AutoOPERATOR option to perform automation for MQSeries events.

Throughout this book, references to OS/390 support also include support for MVS and z/OS.

MAINVIEW AutoOPERATOR and MAINVIEW AutoOPERATOR for MQSeries are sometimes referred to as AutoOPERATOR and AutoOPERATOR for MQSeries, respectively.

How This Book Is Organized

This book is organized as follows. In addition, this book contains an index.

Chapter/Appendix	Description
Chapter 1, "What MAINVIEW AutoOPERATOR for MQSeries Is"	describes AutoOPERATOR for MQSeries and introduces basic terms and concepts key to understanding fundamental MQSeries functions
Chapter 2, "Implementing MAINVIEW AutoOPERATOR for MQSeries"	describes what needs to be done to implement AutoOPERATOR for MQSeries (for example, specifying product option password keys, what software levels are required, and how to define communications to OS/390 and non-OS/390 queue managers)
Chapter 3, "Customizing MAINVIEW AutoOPERATOR for MQSeries"	describes BBPARM member parameters that determine how AutoOPERATOR for MQSeries operates on your system and how to specify queue managers that will be monitored by AutoOPERATOR for automation

Chapter/Appendix	Description
Chapter 4, "Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics"	describes how to view automation statistics displayed by the MAINVIEW AutoOPERATOR for MQSeries Workstation
Chapter 5, "Automating MQSeries Events"	describes how AutoOPERATOR for MQSeries automates MQSeries events with Rules and provides two examples that show step-by-step how to create an MQS Rule
Chapter 6, "Using MAINVIEW AutoOPERATOR for MQSeries Solutions"	describes how to set up, invoke, and disable the MAINVIEW AutoOPERATOR for MQSeries solutions
Chapter 7, "Command MQ Automation Power Line (APL)"	describes the Command MQ Automation Power Line (QMQPOWER) and its parameters
Chapter 8, "Securing MAINVIEW AutoOPERATOR for MQSeries"	describes some security considerations for AutoOPERATOR for MQSeries
Chapter 9, "Using MAINVIEW AutoOPERATOR for MQSeries IMFEXEC Statements"	describes the AutoOPERATOR MQI EXEC interface commands and several other IMFEXEC command statements that allow you to interface with MQSeries through AutoOPERATOR EXECs
Appendix A, "Diagnosing MAINVIEW AutoOPERATOR for MQSeries Errors"	describes resources and procedures to help resolve problems that you may encounter while installing or using MAINVIEW AutoOPERATOR for MQSeries
Appendix B, "MQI Sample Coding"	describes sample code that contains all the pieces necessary to execute an MQSeries request and can be copied into your EXECs and modified to fit your needs
Appendix C, "Rules Variables"	describes all of the variables – Rules and Rule scheduled EXECs
Appendix D, "EXECs Variables"	describes all of the EXEC variables
Appendix E, "Conversion Checklist"	describes the difference between Rules in AutoOPERATOR 5.1 and the Rules in MAINVIEW AutoOPERATOR 6.2

Related Documentation

BMC Software products are supported by several types of documentation:

- online and printed books
- online Help
- release notes and other notices

Note: MAINVIEW AutoOPERATOR also provides online message information. For details about how to view the messages that MAINVIEW AutoOPERATOR generates, see Appendix A, "Diagnosing MAINVIEW AutoOPERATOR for MQSeries Errors".

In addition to this book and the Help, you can find useful information in the publications listed in the following table. As “Online and Printed Books” explains, these publications are available on request from BMC Software.

Document	Description
<i>MAINVIEW AutoOPERATOR Customization Guide</i>	contains procedures for customizing the MAINVIEW AutoOPERATOR product to your site's needs
<i>MAINVIEW AutoOPERATOR Basic Automation Guide</i>	contains procedures for performing basic automation tasks in the data center
<i>MAINVIEW AutoOPERATOR Advanced Automation Guide</i>	contains procedures for using TSO CLIST and REXX EXECs with MAINVIEW AutoOPERATOR to create EXECs that you can use to automate your environment
<i>MAINVIEW Common Customization Guide</i>	contains the instructions for setting up the operational environment of all MAINVIEW products to your site's requirements
<i>MAINVIEW Installation Requirements Guide</i>	contains prerequisite information that is required for the installation of MAINVIEW products on OS/390 and z/OS systems
<i>MAINVIEW Administration Guide</i>	contains the instructions for maintaining and managing the operational environment of all MAINVIEW® products installed at your site
<i>Implementing Security for MAINVIEW Products</i>	describes how to implement MAINVIEW security with your external security manager (ESM) to protect MAINVIEW product resources from user access
<i>Using MAINVIEW</i>	describes how to use MAINVIEW
<i>OS/390 and z/OS Installer Guide</i>	contains detailed information about the OS/390 and z/OS Installer
<i>MAINVIEW Alarm Manager User Guide</i>	describes how to use MAINVIEW® Alarm Manager
<i>MAINVIEW Products General Information</i>	provides an overview of the MAINVIEW environment and the products that it supports

Online and Printed Books

The books that accompany BMC Software products are available in online and printed formats. Online books are formatted as Portable Document Format (PDF) files. Some online books are also formatted as HTML files.

To Access Online Books

To view any online book that BMC Software offers, visit the Customer Support page of the BMC Software Web site at <http://www.bmc.com/support.html>. You can also access PDF books from the documentation compact disc (CD) that accompanies your product.

Use the free Acrobat Reader from Adobe Systems to view, print, or copy PDF files. In some cases, installing the Acrobat Reader and downloading the online books is an optional part of the product-installation process. For information about downloading the free reader from the Web, go to the Adobe Systems site at <http://www.adobe.com>.

To Request Additional Printed Books

BMC Software provides some printed books with your product order. To request additional books, go to <http://www.bmc.com/support.html>.

Other Related Documentation

- *IBM MQSeries Application Programming Guide*, SC33-0807
- *IBM MQSeries for Application Programming Reference*, SC33-1673
- *IBM MQSeries Programmable System Management*, SC33-1482
- *MQSeries for Windows NT and Windows 2000 V5R2-1 Quick Beginnings*, GC34-5389
- *IBM MQSeries Command Reference*, SC33-1369
- *IBM MQSeries for OS/390 Messages and Codes*, GC34-5891

There are many IBM publications for the IBM MQSeries set of products. BMC Software recommends that you review these IBM publications before you attempt to use the MAINVIEW AutoOPERATOR MQI EXEC interface:

- *IBM MQSeries Application Programming Guide*, SC33-0807
- *IBM MQSeries Application Programming Reference*, SC33-1212

For more information about other available MQSeries publications, the section “About this book” in almost any IBM MQSeries book contains a complete list of MQSeries publications.

Online Help

The MAINVIEW AutoOPERATOR product includes Help. In the MAINVIEW AutoOPERATOR ISPF interface, you can access Help by pressing **F1** from any ISPF panel.

Release Notes and Other Notices

Printed release notes accompany each BMC Software product. Release notes provide current information such as

- updates to the installation instructions
- last-minute product information

In addition, BMC Software sometimes provides updated product information between releases (in the form of a flash or a technical bulletin, for example). The latest versions of the release notes and other notices are available on the Web at <http://www.bmc.com/support.html>.

Conventions

This section provides examples of the conventions used in this book and explains how to read ISPF panel-flow diagrams and syntax statements.

General Conventions

This book uses the following general conventions:

Item	Example
information that you are instructed to type	Type SEARCH DB in the designated field. Type search db in the designated field. (Unix)
specific (standard) keyboard key names	Press Enter .
directories, file names, Web addresses, e-mail addresses	The BMC Software home page is at www.bmc.com .
code examples, syntax statements, system messages, screen text	//STEPLIB DD The table <i>tableName</i> is not available.

Item	Example
emphasized words, new terms	The instructions that you give to the software are called <i>commands</i> .
variables	In this message, the variable <i>fileName</i> represents the file that caused the error.

This book uses the following types of special text:

Note: Notes contain important information that you should consider.

Warning! Warnings alert you to situations that could cause problems, such as loss of data, if you do not follow instructions carefully.

Syntax Statements

Syntax statements appear in Courier. The following example shows a sample syntax statement:

```
COMMAND KEYWORD1 [KEYWORD2 |KEYWORD3] KEYWORD4={ YES |NO }
      fileName...
```

The following table explains conventions for syntax statements and provides examples:

Item	Example
Brackets indicate a group of options. You can choose at least one of the items in the group, but none of them is required. Do not type the brackets when you enter the option. A comma means that you can choose one or more of the listed options. You must use a comma to separate the options if you choose more than one option.	[<i>tableName, columnName, field</i>]
Braces enclose a list of required items. You must enter at least one of the items. Do not type the braces when you enter the item.	{ <i>DBDName tableName</i> }
A vertical bar means that you can choose only one of the listed items. In the example, you would choose either <i>commit</i> or <i>cancel</i> .	{ <i>commit cancel</i> }

The following guidelines provide additional information about syntax statements:

- Read diagrams from left to right and from top to bottom.
- A recursive (left-pointing) arrow above a stack indicates that you may choose more than one item in the stack.
- An underlined item is a default option.
- If an example shows punctuation marks, parentheses, or similar symbols, you must enter them as part of the syntax.
- In general, MVS commands, keywords, clauses, and data types are displayed in uppercase letters. However, if an item can be shortened, the minimum portion of the MVS command or keyword might be displayed in uppercase letters with the remainder of the word in lowercase letters (for example, CANcel).
- Items in lowercase letters are values you supply.



Chapter 1 What MAINVIEW AutoOPERATOR for MQSeries Is

This chapter provides an introduction to terms and concepts used for the IBM product MQSeries and the BMC Software product component MAINVIEW AutoOPERATOR for MQSeries.

Understanding MAINVIEW AutoOPERATOR for MQSeries

MAINVIEW AutoOPERATOR for MQSeries is a password key-activated MAINVIEW AutoOPERATOR product component that allows you to automate MQSeries operations and system management.

The IBM product MQSeries uses the concept of *messaging and queuing*, where a message is a collection of data or information sent by one application to another. The message can be sent to a different computer where it resides in a queue until it is retrieved by the other application. The computer can be in a remote location and it can even use a different platform.

MAINVIEW AutoOPERATOR for MQSeries can *listen* for MQSeries events and provide automated responses to the events. MAINVIEW AutoOPERATOR for MQSeries also provides an EXEC interface where you can issue MQSeries commands and receive responses with a set of IMFEXEC statements created especially for MQSeries.

Understanding Automation for MQSeries Events

The following lists examples where MAINVIEW AutoOPERATOR Rules can be used to automate MQSeries events:

- Generate a MAINVIEW AutoOPERATOR ALERT whenever a message is put on the dead letter queue.
- Offload messages when a queue has reached its threshold
- Recover MQSeries channels when a failure occurs.
- Generate a command that will move messages from a system queue which has reached its capacity to an OS/390 generation data set.
- Issue an ALERT to notify operations when messages that are set to be received at a specified rate are not received at that rate.
- Issue an ALERT to notify operations whenever the queue size reaches a given threshold.

MAINVIEW AutoOPERATOR provides solutions that you can immediately use for these situations. For more information, refer to Chapter 6, “Using MAINVIEW AutoOPERATOR for MQSeries Solutions”.

Understanding Terms and Concepts

This section describes the terms and concepts used for MQSeries and MAINVIEW AutoOPERATOR for MQSeries.

What Messaging and Queuing Is

Messaging and queuing allows programs to communicate with each other across a network. With messaging and queuing there is no need for private, dedicated, or logical connections to link the programs. Programs can communicate by putting messages on queues and taking messages from queues. The terms *Message* and *Queue* are described in the following paragraphs.

Message—A message is some data or other communication which a program (or application) sends to a queue for another application to use. A message can be as big as 100 megabytes (MB), although not all platforms support messages of this size.

Queue—A queue is a type of MQSeries object which stores the message until it can be retrieved by another application. If you are running MQSeries for OS/390 5.2 and using shared queues, messages can reside in the OS/390 Coupling Facility. A queue can exist on the same system as the sending and receiving applications. This is called a local queue. Queues that exist on a separate system (or in another LPAR) are called remote queues. A queue name can be up to 48 characters long.

One of the advantages of using MQSeries for messaging and queuing is that, once an application sends a message to a queue, the program can continue to run because the message is safely stored in the queue. The retrieving application can collect the message whenever it is ready. In other words, messaging and queuing allows applications to operate independently of each other while still passing data back and forth at different speeds and times, to and from different locations, and without requiring a dialog between applications.

What a Queue Manager Is

In MQSeries, queues on individual systems are managed by a component called a queue manager. The queue manager provides messaging and queuing services to applications. The queue manager's job is to make sure messages are either put on the correct queue or delivered to another queue manager to accomplish the job. Applications do this through Message Queue Interface (MQI) statements to the queue manager.

On certain platforms (such as OS/390) you can have more than one queue manager per system. On other platforms, you may be restricted to only one queue manager per system. To learn more about platform restrictions, refer to the correspondent platform-specific IBM MQSeries manual.

The following describes different types of queues and queue managers.

Local Queue Manager

A local queue manager is a queue manager that is on the same system as the connecting application. For example, in MQSeries for OS/390, the queue manager can be identified by the subsystem ID in the IEFSSN member of SYS1.PARMLIB. On OS/390 platforms, there may be more than one local queue manager on an OS/390 image.

Remote Queue Manager

A remote queue manager is a queue manager other than the queue manager to which the application is currently connected.

Dead Letter Queue

A dead letter queue is a queue where a queue manager (or application) delivers messages when it cannot deliver the message to its correct destination.

Transmission Queue

A transmission queue is a local queue (to which a channel is connected) where messages that are targeted to a remote queue are temporarily stored.

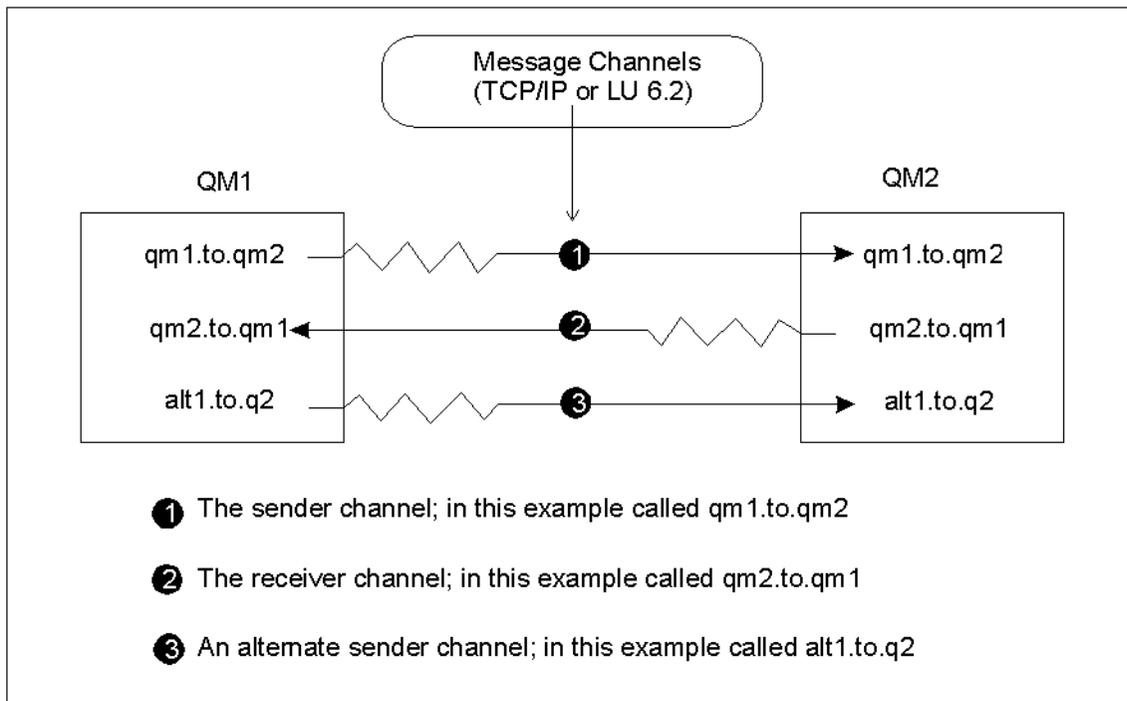
How Queue Managers Communicate

The following term is used to describe connections between queue managers when using distributed queuing:

- Channel (Also Known As Message Channel)

A channel is a mechanism for moving messages from one queue manager to another. A message channel is comprised of two agents (a sender and a receiver) and a communication link. A message channel agent is a program that sends the message from the transmission queue to (or from) a communication link and then to the destination queue.

Figure 1-1 Example of Channels



What MQSeries Events Are

MAINVIEW AutoOPERATOR can listen for events, such as OS/390 commands or messages, and fire off Rules for that event which perform an action, such as message rerouting or creating a MAINVIEW AutoOPERATOR ALERT. Messaging and queuing also generates MQSeries instrumentation events for which MAINVIEW AutoOPERATOR for MQSeries can listen and fire off Rules.

MAINVIEW AutoOPERATOR distinguishes MQSeries messages based on the message format:

- MQFMT_NONE
- MQFMT_ADMIN
- MQFMT_CHANNEL_COMPLETED
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING

- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- USER (Used for any format except MQFMT_EVENT)

When you create a Rule, you must specify which format the message contains, or specify USER for any format except MQFMT_EVENT. A single Rule can recognize *only one* of these two types.

Messages with Format MQFMT_EVENT

Messages with format MQFMT_EVENT are the instrumentation events provided by MQSeries. In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. The result of such an event is that the queue manager or channel instance puts a special message, called an instrumentation event message, on an event queue.

MAINVIEW AutoOPERATOR for MQSeries extracts information appropriate for automation from instrumentation event messages and makes it available to Rules and EXECs in variables. For more information, refer to Chapter 5, “Automating MQSeries Events”.

There are three categories of MQSeries instrumentation events, and seven types of events within the three categories.

Queue manager events This category of instrumentation events is 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 are the event types within the queue manager instrumentation event category, and their associated queue manager attributes:

Authority (AUTHOREV)

Inhibit (INHIBTEV)

Local (LOCALEV)

Remote (REMOTEEV)

Start and Stop (STRSTPEV)

For each event type in this list, the associated queue manager attribute enables or disables instrumentation events of that type. For more information about enabling and disabling instrumentation event types, see the IBM publication *MQSeries Command Reference*.

Performance events This category of instrumentation events is a notification that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached.

When a performance event is generated, the queue manager puts the associated event message on the SYSTEM.ADMIN.PERFM.EVENT queue.

All events in the performance event category fall into the performance type.

The queue manager attribute PERFMEEV enables or disables the performance event type. For more information about enabling and disabling instrumentation event types, see the IBM publication *MQSeries Command Reference*.

Channel events This category of instrumentation events is reported by channels as a result of conditions detected during their operation; for example, when a channel instance is stopped.

When a channel event is generated, the queue manager puts the associated event message on the SYSTEM.ADMIN.CHANNEL.EVENT queue.

All events in the channel event category fall into the channel type.

Most of the events in the channel event category are enabled automatically and cannot be enabled or disabled by command. The exceptions are the two automatic channel definition events. The queue manager attribute CHADEV enables or disables these two channel events. For more information about enabling and disabling instrumentation event types, see the IBM publication *MQSeries Command Reference*.

For more information about instrumentation events, refer to the IBM publication *MQSeries Programmable System Management*.

Messages with Format Other Than MQFMT_EVENT

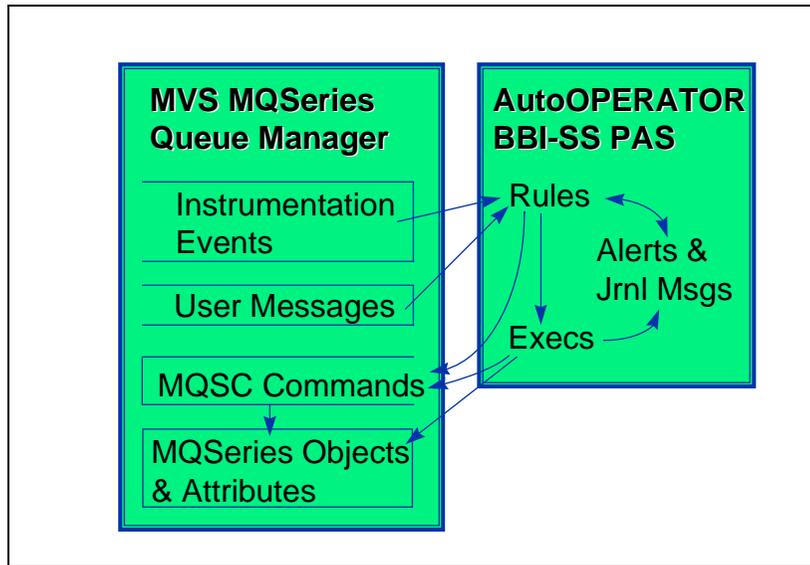
Non-MQFMT_EVENT messages are communications sent from one user-written program to another. Several IBM program products use this capability. MAINVIEW AutoOPERATOR for MQSeries can also utilize Non-MQFMT_EVENT messages. For example, you could have a payroll application that sends data to and from MQSeries user queues. You can create Rules to fire based on the data (or message) that is received.

Interactions between MAINVIEW AutoOPERATOR and MQSeries

MAINVIEW AutoOPERATOR automates both OS/390 and non-OS/390 MQSeries queue managers and responds to the MQS events. The diagrams in this section illustrate the relationships between the components of MAINVIEW AutoOPERATOR and both OS/390 and non-OS/390 queue managers.

Figure 1-2 on page 1-9 shows the interaction between MAINVIEW AutoOPERATOR and an OS/390 queue manager.

Figure 1-2 Interaction between MAINVIEW AutoOPERATOR and an OS/390 Queue Manager



MAINVIEW AutoOPERATOR is connected to an OS/390 MQSeries queue manager, which it listens to for instrumentation events and non-event messages. If the instrumentation event or non-event message is set to trigger a Rule for that event, the Rule can perform one or more of the following actions:

- create an ALERT or journal message
- launch an EXEC
- issue a MQSeries command to manipulate MQSeries objects and attributes

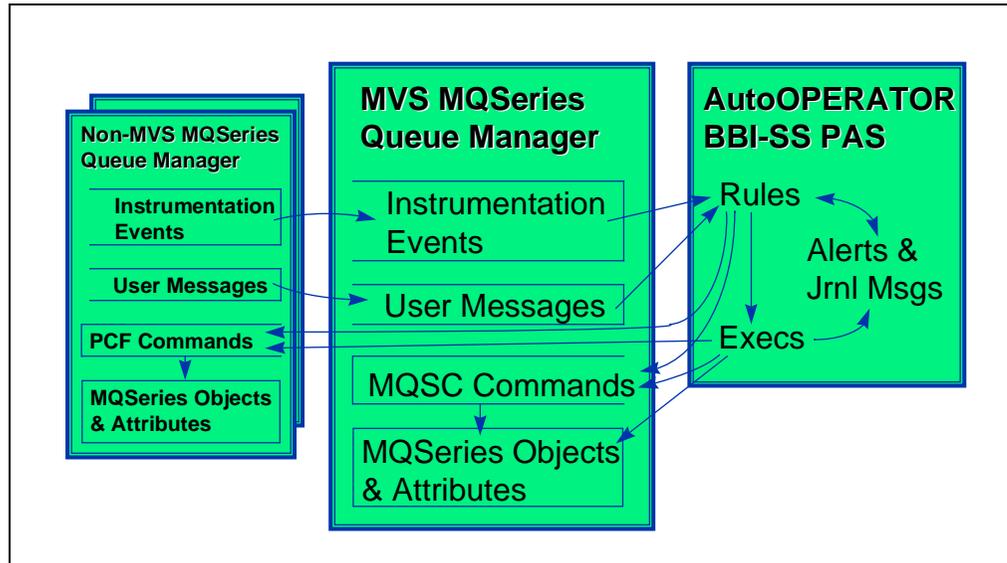
If the Rule issues an ALERT, the ALERT is sent to an operator who performs the appropriate action. A journal message is also logged in the BBI Journal, which contains messages about the queue manager and queues.

If an EXEC is launched, it can create an ALERT, issue a MQSeries command to alter a MQSeries object or attribute, or get and put messages directly using the MQI. The EXEC may also log a message to the BBI Journal.

If the Rule or the EXEC issues a MQSeries command, the MQSeries objects and attributes can be modified to resolve the issue that initially triggered the MQS event.

Figure 1-3 illustrates the relationship between MAINVIEW AutoOPERATOR and both an OS/390 and non-OS/390 (remote) queue manager.

Figure 1-3 Relationship between MAINVIEW AutoOPERATOR and an OS/390 and Non-OS/390 Queue Manager



MAINVIEW AutoOPERATOR interacts with a non-OS/390 queue manager in the following ways:

- The non-OS/390 queue manager sends its instrumentation events and non-event messages through the OS/390 queue manager to be listened to by MAINVIEW AutoOPERATOR.
- MAINVIEW AutoOPERATOR Rules and EXECs can send queue manager commands to the non-OS/390 queue manager in response to these messages and events.
- MAINVIEW AutoOPERATOR Rules or EXECs can issue a PCF (Programmable Command Format) command to manipulate the MQSeries objects and attributes in the non-OS/390 queue manager.

Chapter 2 Implementing MAINVIEW AutoOPERATOR for MQSeries

This chapter describes what you need to do to implement MAINVIEW AutoOPERATOR for MQSeries.

How AutoOPERATOR Connects to MQSeries

For MAINVIEW AutoOPERATOR for MQSeries to perform automation of MQSeries events, AutoOPERATOR must make connections to MQSeries local queue managers. MAINVIEW AutoOPERATOR for MQSeries determines which queue managers to connect with automatically and dynamically based on automation needs.

MAINVIEW AutoOPERATOR for MQSeries can connect directly only to MQSeries queue managers running on the same OS/390 image. However, MQSeries instrumentation events on non-OS/390 operating systems can still be automated by MAINVIEW AutoOPERATOR for MQSeries if the non-OS/390 is connected to an OS/390 queue manager. Refer to “Step 3. Defining Connectivity” on page 2-14 for more information.

Connections between OS/390 queue managers and AutoOPERATOR can be made in one of two ways:

1. For Rules created for event type MQS, AutoOPERATOR searches for the queue manager during Rule enablement. If the queue manager is found and a connection has not already been established, a connection is made.

2. When a queue manager completes initialization, AutoOPERATOR checks for any enabled Rules created for event type MQS that require a connection and if any are found, a connection is made.

At the time of connection, *only* the eligible queue names listed in BBPARM member AAOMQLxx become eligible for automation. If new queues are added later that require automation, you must refresh the list of eligible queues. For more information about including specific queue managers in BBPARM member AAOMQLxx, refer to “Parameters for BBPARM Member AAOMQLxx” on page 3-5.

Note: At MAINVIEW AutoOPERATOR startup time or during a reset situation, the MAINVIEW AutoOPERATOR for MQSeries creates the following MQSeries queues for internal processing of commands and replies:

- BBMVAO.COMMAND.REPLY.MODEL
- BBOMVAO.xxxx.RULES.INITIALIZE
- BBOMVAO.xxxx.RULES.CMDREPLY
- BBOMVAO.EXEC.REPLY.xxxx.yyyy

where xxxx is the MAINVIEW AutoOPERATOR subsystem ID and yyyy is the name of the queue manager

You should not be concerned with these queues or events related to these queues unless error messages are issued regarding them.

Required Software

The MAINVIEW AutoOPERATOR for MQSeries component requires a minimum of MAINVIEW AutoOPERATOR Version 6, Release 2, Modification 0.

At least one of the following is also required:

- MQSeries for OS/390 Version 1, Release 1, Modification 3
- MQSeries for OS/390 Version 1, Release 1, Modification 4
- MQSeries for OS/390 Version 1, Release 2, Modification 0
- MQSeries for OS/390 Version 2, Release 1, Modification 0
- MQSeries for OS/390 Version 5, Release 2, Modification 0

If you want MAINVIEW AutoOPERATOR for MQSeries to automate MQSeries activity on other operating systems, you need one or more of the following MQSeries products:

- MQSeries for UNIX
- MQSeries for OS/2 Version 2.0.1 or later
- MQSeries for OS/400 Version 3 Release 1 or later
- MQSeries for Windows NT

Note: For MAINVIEW AutoOPERATOR for MQSeries to automate MQS events from non-OS/390 event queues, *you must verify that the connectivity between OS/390 and non-OS/390 queue managers is defined*, as described in “Step 3. Defining Connectivity” on page 2-14.

If you are installing Command MQ On Ramp, you need the following products:

- A Web browser that runs the JDK 1.02 level of Java, such as Netscape Navigator 3.04 or Internet Explorer 3.0

Internet Explorer 4.03 does not run Java correctly.

Problems are only supported if they can be reproduced under Netscape Navigator 3.04.

- OS/390 TCP/IP 3.1 or later installed on your OS/390 system

This release of TCP/IP is a base component of OS/390 1.1 and runs on any level of OS/390 currently supported by AutoOPERATOR.

- One PATROL[®] Node Manager for MQ - Administrator product for each non-OS/390 target platform to which you will connect

For more information about this product, refer to the BMC Software documents *PATROL for WebSphere MQ Installation Guide* and *PATROL for WebSphere MQ Administration Guide*.

Installation Overview

This table shows an overview of the MAINVIEW AutoOPERATOR for MQSeries installation activities.

Table 2-1 Installation Activities — Summary (Part 1 of 2)

Summary of Steps	For More Information, See
If you are an existing MAINVIEW AutoOPERATOR for MQSeries site, review the migration considerations.	"Installation Overview" on page 2-4
Enable MAINVIEW AutoOPERATOR for MQSeries by adding the appropriate Product Option Password Keys.	"Specifying Product Option Password Keys" on page 2-6
Add MQSeries APF-authorized libraries to MAINVIEW AutoOPERATOR STEPLIB.	"Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card" on page 2-8
<p>(optional)</p> <ul style="list-style-type: none"> • Activate automatic enabling of MQSeries events • Add MQEV=YES to configuration member AAOPRM00. 	<ul style="list-style-type: none"> • "Modifying AAOPRM00 Parameter Settings" on page 2-8, • "Step 2. Enabling Instrumentation Events" on page 2-13 • Chapter 3, "Customizing MAINVIEW AutoOPERATOR for MQSeries"
Add BQ to the RULESET= list in AAOPRM00.	"Modifying AAOPRM00 Parameter Settings" on page 2-8
<ul style="list-style-type: none"> • Modify configuration member AAOMQL00. • Include queue names against which MQSeries Rules are written. 	<ul style="list-style-type: none"> • "Modifying AAOMQL00 Parameter Settings (Optional)" on page 2-9 • Chapter 3, "Customizing MAINVIEW AutoOPERATOR for MQSeries"
(optional) Enable distributed diagnostic Rules.	"Enabling Distributed Diagnostic Rules" on page 2-11
Review security.	<ul style="list-style-type: none"> • "MAINVIEW AutoOPERATOR for MQSeries Security Considerations" on page 2-12 • "Securing MAINVIEW AutoOPERATOR for MQSeries" on page 8-1
Enable MQSeries instrumentation events on the OS/390 queue manager that you want monitored.	<ul style="list-style-type: none"> • "Modifying AAOPRM00 Parameter Settings" on page 2-8, • "Enabling Instrumentation Events for OS/390 Queue Managers" on page 2-14, • "Generating Instrumentation Events" on page 3-3, • "Chapter 3 Customizing MAINVIEW AutoOPERATOR for MQSeries," • "Enabling MQS Events" on page 5-5

Table 2-1 Installation Activities — Summary (Part 2 of 2)

Summary of Steps	For More Information, See
<i>(optional)</i> Enable MQSeries instrumentation events on non-OS/390 queue manager.	<ul style="list-style-type: none"> • “Step 2. Enabling Instrumentation Events” on page 2-13 • “Step 3. Defining Connectivity” on page 2-14
<p>If PATROL® for MQ-Administrator or PATROL® for MQ-Operator is installed on the OS/390 system, you need to do the following steps to allow MAINVIEW AutoOPERATOR for MQSeries to co-exist with these products:</p> <ul style="list-style-type: none"> • Modify member AAOMQL00. • Modify existing rules to use the new event queue name. 	“Step 5. Setting Up MAINVIEW AutoOPERATOR for MQSeries to Co-Exist with PATROL for WebSphere MQ” on page 2-31
<i>(optional)</i> Define sender/receiver channel pairs between OS/390 MQ queue managers and non-OS/390 queue managers. This allows MQ to transmit messages and commands between the queue managers.	“Step 3. Defining Connectivity” on page 2-14
<i>(optional)</i> Define queue SYSBMC.DISTRIB.EVENTS for non-OS/390 events and include this queue in UBBPARAM(AAOMQL00).	<ul style="list-style-type: none"> • “Modifying AAOMQL00 Parameter Settings (Optional)” on page 2-9 • “Define Queue for Non-OS/390 Instrumentation Events” on page 2-13
<i>(optional)</i> Route non-OS/390 instrumentation events to SYSBMC.DISTRIB.EVENTS on an OS/390 MQ queue manager to make the events available to MAINVIEW AutoOPERATOR for MQSeries.	“Step 3. Defining Connectivity” on page 2-14
<i>(optional)</i> Set Up MQ command capability for non-OS/390 MQ queue managers.	“Step 3. Defining Connectivity” on page 2-14
If MQGINHIB=ALTER or MQNSHARE=ALTER, verify that all queues to be monitored have the correct attributes.	Chapter 3, “Customizing MAINVIEW AutoOPERATOR for MQSeries”
IVP: verify that MAINVIEW AutoOPERATOR for MQSeries is correctly customized.	“Step 4. Using the Installation Verification Procedure” on page 2-27
Implement MAINVIEW AutoOPERATOR for MQSeries solutions.	Chapter 6, “Using MAINVIEW AutoOPERATOR for MQSeries Solutions”
If SAAREXX/370 is not installed, install REXX/370 alternate library.	Chapter 6, “Using MAINVIEW AutoOPERATOR for MQSeries Solutions”
Create site-specific MQ automation Rules to capture MQ messages as MAINVIEW AutoOPERATOR events.	<ul style="list-style-type: none"> • Chapter 5, “Automating MQSeries Events” • Appendix C, “Rules Variables”
Create EXECs to manipulate MQ messages.	<ul style="list-style-type: none"> • Chapter 9, “Using MAINVIEW AutoOPERATOR for MQSeries IMFEXEC Statements” • Appendix D, “EXECs Variables”
<i>(optional)</i> Create EXECs to manage and maintain non-OS/390 MQ queue managers.	Chapter 7, “Command MQ Automation Power Line (APL)”
<i>(optional)</i> If you are still using release 5.1 and want to take advantage of new features and message data, convert EXECs replacing the old 5.1 statements with the MQI statements that became available in 6.1.	Appendix E, “Conversion Checklist”

Step 1. Activating MAINVIEW AutoOPERATOR for MQSeries

This step describes the initial procedures you must perform to activate AutoOPERATOR for MQSeries.

Specifying Product Option Password Keys

AutoOPERATOR has features and options that require password keys based upon information about your CPU. BMC Software supplies a utility program called SYSINFO to extract the CPU information.

To extract the CPU information, follow these steps:

- Step 1** Run the SYSINFO utility to obtain system information for product password keys.

On the OS/390 system where these products operates, type

TSO CALL 'hilevel.BBLINK(SYSINFO)'

where *hilevel* represents your high-level data set qualifier.

- Step 2** Write down the output from the SYSINFO utility in the format that follows:

```
CPU Type_ _ _ _ _
CPU Model_ _
CPU Serial Number_ _ _ _ _
```

- Step 3** Give this output information to your BMC Software representative, who will give you your password key.

- Step 4** Edit BBPARM member BBKEYS (create one if you do not have one), and add the key parameters.

If you receive more than one key, each key must begin on a separate line. During key validation, AutoOPERATOR attempts to use the most suitable key by validating each key and selecting the one most qualified for your CPU.

Specify

KEY=ppp-cccc-tt-nnnnn-yyddd-xxxx[-ssss]

where

ppp Is a three-character product ID; such as

ANV MAINVIEW AutoOPERATOR Access NV
 CAO MAINVIEW AutoOPERATOR for CICS
 AO MAINVIEW AutoOPERATOR for IMS
 MAO MAINVIEW AutoOPERATOR for OS/390
 QAO MAINVIEW AutoOPERATOR for MQSeries
 TSH MAINVIEW AutoOPERATOR TapeSHARE

To use the MAINVIEW AutoOPERATOR TapeSHARE option, MAINVIEW AutoOPERATOR for OS/390 must be installed and the MAO key must be specified. Refer to the *MAINVIEW AutoOPERATOR Customization Guide* for more information.

cccc is a four-character CPU type

tt is a two-character CPU internal model group type

nnnnn is a four- to five-digit CPU serial number in hexadecimal format

yyddd is the julian expiration date

xxxx is a four-character authorization key created by BMC Software

Note: The values for *ppp*, *cccc*, *tt*, or *nnnnn* can be a generic * qualifier; however, the Product Key created by BMC Software must have been generated using this same qualifier.

ssss is the SMF ID of the system where the product option should be activated

The use of *ssss* helps prevent an IPL failure because it identifies an invalid product password key and allows for immediate correction. When an invalid key is detected, a WTOR message is issued and the operator can specify another key immediately. If the SMF ID of the target system is not specified, no external warning messages are issued for invalid keys.

Note: If the CPU serial number for your site changes or an additional CPU is added, BMC Software must receive the new number so that a new Product Key can be generated.

Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card

If the MQSeries libraries are not in the Link List, add the following MQSeries APF-authorized program library or libraries to the STEPLIB DD card in the BBI-SS PAS job:

prefix.SCSQAUTH
prefix.SCSQANLE (for MQSeries for ESA v1.1.4 and later)

where *prefix* was defined when you installed MQSeries.

Note: If you are running more than one queue manager on OS/390, use the MQSeries libraries for the highest level of MQSeries.

Modifying AAOPRM00 Parameter Settings

In BBPARM member AAOPRM00, make the following changes:

1. Specify the setting for the parameter MQEV= as

MQEV=YES

MQEV=YES specifies that MAINVIEW AutoOPERATOR for MQSeries should automatically enable instrumentation events for a queue manager during connection if they are not enabled already.

If the parameter MQEV= is not specified, the default is MQEV=NO.

2. Update the RULESET= parameter to include the AAORUL00 Rule Set suffix BQ; for example,

RULESET=(BC,CM,BQ)

This ensures that the Rule Set AAORULBQ is enabled when the BBI-SS PAS is cold-started.

For information about other parameters in BBPARM member AAOPRM00, refer to “Parameters for BBPARM Member AAOPRMxx” on page 3-1.

Modifying AAOMQL00 Parameter Settings (Optional)

Review the parameter settings in BBPARM member AAOMQL00. The parameters in this member specify which MQSeries queues can be monitored (or listened to) by MAINVIEW AutoOPERATOR for MQSeries automation. The parameters are as follows:

- TYPE=(INCL|EXCL)

Specifies whether MAINVIEW AutoOPERATOR for MQSeries should include or exclude this queue in the set of queues that is eligible for automation. Possible values are:

INCL	Specifies that AutoOPERATOR should make this queue eligible for possible automation. This is the default setting.
EXCL	Specifies that AutoOPERATOR should not make this queue eligible for automation.

The abbreviation for TYPE is T and its values can be abbreviated from INCL to I and from EXCL to E. For example, T(I) or T(E).

- QMGR(queue manager name)

Specifies the four-character ID for a local OS/390 queue manager that AutoOPERATOR will monitor. You must specify a queue manager name or a partial queue manager name. There is no default value for this parameter.

The abbreviation for QMGR is M.

For example, M(CSQ1).

- QUEUE(queue name)

Specifies the 48-character name of the queue that AutoOPERATOR will make eligible for automation. You must specify a queue name or a partial queue name. There is no default value for this parameter.

The abbreviation for QUEUE is U.

For example, U(SYSTEM.ADMIN.QMGR.EVENT)

- OPEN

A keyword that gives you control over how the queue is opened and what to do with the messages in the queue after it is opened. The abbreviation for OPEN is O. Its subparameters are as follows.

Option 1:

- EXCLUSIVE indicates the queue should be opened with the MQOO_INPUT_EXCLUSIVE open option. This means other applications are unable to open the queue while MAINVIEW AutoOPERATOR has it open. It also means that if another application has the queue opened, MAINVIEW AutoOPERATOR will be unable to open it. Its abbreviation is E.
- SHARED indicates the queue should be opened with the MQOO_INPUT_SHARED option. This subparameter is the default for option 1. Its abbreviation is S.

Option 2:

- PROCESSOLD indicates that all messages found on the queue when it is opened should be routed through the Rule Processor to allow automation to take place. This parameter is useful for processing messages that were put on the queue while the queue was not opened by MAINVIEW AutoOPERATOR. The abbreviation for this option is P.
- IGNOREOLD indicates that MAINVIEW AutoOPERATOR will not process the existing messages on the queue and no automation takes place for those messages. Its abbreviation is I. This subparameter is the default for option 2.

Example

Example 1: OPEN(EXCLUSIVE,PROCESSOLD) or O(E,P)

Indicates that the queue is to be opened with the MQOO_INPUT_EXCLUSIVE option, and that all existing messages found on the queue at open time should be routed through the Rule Processor for automation.

Example

Example 2: OPEN(SHARED,IGNOREOLD) OR O(S,I)

Indicates that the queue is to be opened with the MQOO_INPUT_SHARED option, and that all existing messages found on the queue at open time should be ignored.

For more information about setting these parameters and an example of how to set them, refer to “Parameters for BBPARM Member AAOMQLxx” on page 3-5.

Refresh the List of Queues Eligible for Automation

To refresh the list of queues eligible for automation, issue the BBI control command:

.RESET MQ xx

Refer to “MAINVIEW AutoOPERATOR for MQSeries BBI Commands” on page 2-33 for more information about BBI commands for MAINVIEW AutoOPERATOR for MQSeries.

Enabling Distributed Diagnostic Rules

Use the AutoOPERATOR Rules Processor application to enable three Rules distributed in BBPARM member AAORULBQ:

- MQDIA001
- MQDIA002
- MQDIA003

These three Rules help detect the most common setup problems, including connectivity and security problems.

Enable each of these three Rules as follows:

1. Schedule the Rule Processor application by typing RULES from any BBI terminal session screen.
2. Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
3. Select the AAORULBQ Rule Set.

4. Select the Rule MQDIA`xxx`.
5. Edit MQDIA`xxx` using the Rules Processor application panels.

Press **Enter** to navigate through the panels and make the following change:

On the Selection Criteria panel under Queue Identification, change `xxx` to the queue manager IDs of the queue managers that this Rule manages. Remember that these queue manager names must be specified in BBPARM member AAOMQL`xx`.

MAINVIEW AutoOPERATOR for MQSeries Security Considerations

You must ensure that AutoOPERATOR is given the necessary permissions to read and write to and from the appropriate MQSeries queues. For example, AutoOPERATOR must be able to read from the queues that you want it to monitor.

To issue commands and receive responses, AutoOPERATOR must be able to put messages into the command queue and read responses from the response queue. AutoOPERATOR dynamically builds queues for command responses and for its solutions. These queues start with the high-level qualifier BBOMVAO. The external security manager must give alter authority for the resource BBOMVAO.* to the BBI-SS PAS. (For more information about security, refer to the appropriate MQSeries system management publication for each operating system.)

For example, if security is active for MQSeries on Microsoft Windows NT, you must define the security ID associated with the AutoOPERATOR BBI-SS PAS to Windows NT and add this security ID to the mqm group. This is required to allow AutoOPERATOR to send commands to MQSeries on Windows NT and retrieve the command response messages. If the commands are issued by AutoOPERATOR users (BBI-TS users) or OS/390 batch jobs, you also need to define these user IDs to Windows NT and add these user IDs to the mqm group.

Users are defined to Windows NT using the User Manager facility. Refer to the publication *IBM MQSeries for Windows NT System Management Guide* and the Windows NT help topic called *User Manager* for additional information about mqm and defining users.

Restarting the BBI-SS PAS

To complete the activation of AutoOPERATOR for MQSeries, you must restart (or bounce) the BBI-SS PAS.

First, you must stop the subsystem; issue the OS/390 STOP command:

```
P BBI-SS_PAS_NAME
```

Then issue the OS/390 START command:

```
S BBI-SS_PAS_NAME
```

Define Queue for Non-OS/390 Instrumentation Events

For each OS/390 MQ/Series Queue Manager that MAINVIEW AutoOPERATOR monitors, define the queue to receive instrumentation events from non-OS/390 queue managers as follows:

```
DEFINE QL(SYSBMC.DISTRIB.EVENTS)  
LIKE(SYSTEM.ADMIN.QMGR.EVENT)
```

Step 2. Enabling Instrumentation Events

This step explains how to enable MQSeries instrumentation events for OS/390 queue managers and non-OS/390 queue managers (if applicable) so that AutoOPERATOR for MQSeries can listen for and automate responses to these events.

You must manually enable instrumentation events for

- all non-OS/390 queue managers
- OS/390 queue managers if you do not choose to set MQEV=YES in AAOPRM00

Enabling Instrumentation Events for Non-OS/390 Queue Managers

There are seven types of non-OS/390 instrumentation events, six of which can be enabled and disabled. The six non-OS/390 instrumentation event types which can be enabled are Author, Inhibit, Local, Performance, Remote, and Start/Stop. The channel seventh event type is always enabled and cannot be disabled. For more information on enabling or disabling the six events, refer to the IBM manual *IBM MQSeries Programmable System Management*.

These events are queued to the instrumentation queues that have limited capacities. MAINVIEW AutoOPERATOR for MQSeries provides a set of solutions to help you manage these queues. Refer to Chapter 6, “Using MAINVIEW AutoOPERATOR for MQSeries Solutions” for more information.

Enabling Instrumentation Events for OS/390 Queue Managers

To specify that MAINVIEW AutoOPERATOR for MQSeries automatically enable instrumentation events for all OS/390 queue managers to which it connects, specify MQEV=YES in BBPARM member AAOPRM00.

For more information about the MQEV= parameter setting, refer to “Modifying AAOPRM00 Parameter Settings” on page 2-8 and “Parameters for BBPARM Member AAOPRMxx” on page 3-1.

Step 3. Defining Connectivity

There are two methods to define intercommunication for OS/390 and non-OS/390 queue managers so that AutoOPERATOR can listen for instrumentation events and issue commands to queue managers:

- Define and activate the queue manager connections using Command MQ On Ramp.

For more information on using Command MQ On Ramp to make connections between OS/390 and non-OS/390 queue managers, refer to “Making Queue Manager Connections Using Command MQ On Ramp” on page 2-15.

For more information about how Command MQ On Ramp diagnoses and repairs connection problems, refer to “Using the Basic Intercommunication Solution” on page 6-24.

- Manually define OS/390 and non-OS/390 queue managers.

To manually configure the queue manager connections for AutoOPERATOR, you must define the queue managers, transmission queues, and sender and receiver channels so AutoOPERATOR can listen for non-OS/390 instrumentation events and issue commands to non-OS/390 queue managers.

For more information on manually connecting OS/390 and non-OS/390 queue managers, refer to “Connecting Queue Managers Manually” on page 2-23.

Making Queue Manager Connections Using Command MQ On Ramp

Command MQ On Ramp is a Web-based application that inspects and optionally completes the connections between an OS/390 queue manager and non-OS/390 queue managers so that Command MQ products can detect MQSeries events and issue MQSeries commands.

Setting Up OS/390 Queue Manager Connectivity

To enable AutoOPERATOR to issue commands to the OS/390 queue manager, you must ensure that the queue manager’s command server is running and the command input queue is usable. Command MQ On Ramp ensures that the command server and its command input queue are operational. For information about the command server and starting it, refer to the IBM manual, *IBM MQSeries Command Reference*.

Setting Up Non-OS/390 Queue Manager Connectivity

MAINVIEW AutoOPERATOR for MQSeries can automate events from non-OS/390 queue managers if the non-OS/390 event messages are directed to local OS/390 queue manager. MAINVIEW AutoOPERATOR for MQSeries is able to detect these non-OS/390 event messages.

Event data is sent to the local OS/390 MQSeries queue manager by subscribing its queue to the PATROL event listener. PATROL automatically transmits the messages from the non-OS/390 queue manager to the OS/390 queue manager. MAINVIEW AutoOPERATOR for MQSeries, monitoring the local queues on the OS/390 queue manager, receives the system event messages from the non-OS/390 queue manager and performs whatever automation has been defined.

The following sections explain how to use Command MQ On Ramp to interconnect queue managers:

- Hyperlinking to Command MQ On Ramp in Your Web Browser
- Accessing Command MQ On Ramp Interface
- Specifying the OS/390 Queue Manager
- Specifying Non-OS/390 Queue Managers
- Completing the Queue Manager Connections
- Logging Off

Hyperlinking to Command MQ On Ramp in Your Web Browser

To Issue Command MQ On Ramp

Step 1 Create a hyperlink in one of your Web pages by adding the following HTML tag:

```
<a href="http://domain name on OS/390:port number/Cor.html">
Welcome to the Command MQ On Ramp</a>
```

where

domain name on OS/390 is the network name or IP address of AutoOPERATOR on OS/390

port number is the TCP/IP port number of AutoOPERATOR Web server

Note: You set AutoOPERATOR port number by setting the LSTNPORT=nnnn parameter in AutoOPERATOR's AAOGMExx member of BBIPARM.

Step 2 Open your Web browser.

Step 3 Open the Web page that you modified in Step 1 from which you want to launch Command MQ On Ramp.

Step 4 Click the Command MQ On Ramp hyperlink.

When you click the hyperlink, a panel appears prompting you to enter a user ID and password, which are the TSO user ID and password on the OS/390 system from which the Command MQ On Ramp applet was loaded.

Step 5 Type your TSO userid and password.

Step 6 Click **Logon** or press Enter.

A status log reports the progress of the applet.

A separate frame opens to run the Command MQ On Ramp session. The browser is free to be used for other purposes, but keep the Command MQ On Ramp page accessible. If you remove the Command MQ On Ramp page from the stack of pages accessible via the Back and Forward buttons in your browser, the browser may terminate the Command MQ On Ramp applet.

Accessing Command MQ On Ramp Interface

Once you log on to Command MQ On Ramp, use the new window to specify the queue manager connections.

The Command MQ On Ramp window provides Next and Logoff buttons for navigation. After completing the appropriate portions of each panel, click Next to activate the selections. To log off at any time, click the Logoff button.

The Command MQ On Ramp window also contains these menu options:

Option	Description
Windows	Allows you to see the following windows: JournalStatus Journal messages logged by the Command MQ On Ramp as it performs its activities. GMEDiagnostic information about the Command MQ On Ramp General Messages Exchange (GME) function. Command MQ On Ramp uses GME to route messages to applications in AutoOPERATOR and to receive messages from them. AppletBrowser window from which you started Command MQ On Ramp.
Help	Describes the version of Command MQ On Ramp you are using.

Specifying the OS/390 Queue Manager

Command MQ On Ramp discovers the names of OS/390 queue managers and displays them in a selection list:

- If you know the name of the OS/390 queue manager that you want to specify, follow these steps:
 1. Select the name from the list.
 2. Click NEXT.
- If the name of the OS/390 queue manager does not appear in the list, follow these steps:
 1. Type the name in the text field.
 2. Click **Next**.

The text box must contain a name in order to proceed.

Specifying Non-OS/390 Queue Managers

When you select an OS/390 queue manager for Command MQ products to use, Command MQ On Ramp prompts you to specify one or more non-OS/390 queue managers to connect to it.

Note: Unlike specifying the OS/390 queue manager, Command MQ On Ramp does not automatically provide a list from which you can select non-OS/390 queue managers. Command MQ On Ramp could discover several hundred non-OS/390 queue managers, and it could take a considerable amount of time to compile the list.

Step 1 Specify a non-OS/390 queue manager:

If you know the name of a queue manager, type the name in the text field.

You can also specify the network address of the queue manager.

Note: When you connect a specified non-OS/390 queue manager to this OS/390 queue manager for the very first time, you must specify its network address.

If you do not know the name of the queue manager:

1.A Click **Discover Non-OS/390 Queue Managers**.

Command MQ On Ramp issues an AutoOPERATOR EXEC to obtain the names and network addresses of the non-OS/390 queue managers already referenced by definitions in the OS/390 queue manager and displays a list of the names discovered.

Note: One PATROL Node for MQ - Administrator product must be installed for each non-OS/390 target platform to which you will connect.

For more information about this product, refer to the BMC Software documents *PATROL for WebSphere MQ Administration Guide* and *PATROL for WebSphere MQ Planning and Implementation Guide*.

1.B From the generated list, select a non-OS/390 queue manager that you want to connect.

The Command MQ On Ramp automatically displays the corresponding network address in the Specify the Network Address text box.

Step 2 Click **Add**.

The non-OS/390 queue manager and its network address appear in the box at the bottom of the screen.

Step 3 Repeat Steps 1 and 2 for each non-OS/390 queue manager.

Step 4 When you are finished adding non-OS/390 queue managers, click **Next**.

Removing a Non-OS/390 Queue Manager

To remove a non-OS/390 queue manager from the list of candidates to be configured, follow these steps:

Step 1 Highlight the name.

Step 2 Click **Delete**.

Step 3 When you are satisfied with the list of non-OS/390 queue managers, click **Next**.

If you want to go back and change the non-OS/390 queue manager, click **Back**.

Completing the Queue Manager Connections

Once OS/390 and non-OS/390 queue managers are specified in the candidate box, the next panel allows you to inspect only or inspect and complete the queue manager connections. If you only inspect the connections, Command MQ On Ramp takes no action to correct any discovered problems. If you inspect and complete the connections, Command MQ On Ramp tries to fix the problem(s) and only reports an error if a solution is not found.

Beneath the list of queue manager connections to be completed is an area that provides the following information about the selected connection:

- Name of the OS/390 queue manager to be connected
- Name of the selected non-OS/390 queue manager to be connected
- Network address of the non-OS/390 queue manager
- Status of the connection

The status continues to change as Command MQ On Ramp inspects and completes each connection.

To check the specified connections, follow these steps:

- Step 1** Determine if the queue manager connections should be only inspected or inspected and completed.
- Step 2** Check either the **Inspect** or the **Complete** radio button.
- Step 3** Click **Inspect Connections** or **Complete Connections**.

Command MQ On Ramp performs the following actions:

- schedules EXECs in AutoOPERATOR to process each connection
- displays a status panel in the Command MQ On Ramp window

The status panel indicates that the connection-checking EXEC is running and whether any connection errors have been detected or repaired.

During the connection process, one of the following colors highlights the queue manager name and indicates the status of the listed queue manager connections.

Color	Diagnosis
White	Connection has not been checked.
Blue	AutoOPERATOR is in the process of checking the connection.
Yellow	Possible connection error was detected.
Red	Connection error was detected.
Green	Connection is complete and ready for use.

If you click the **Back** button at the bottom of the status panel, Command MQ On Ramp stops scheduling the configuration EXEC. Clicking the **Back** button also returns you to the Specify the non-OS/390 Queue Manager(s) configure window. You can select additional queue managers to be configured and validated.

When you click Next and resume interconnection checking, Command MQ On Ramp checks only newly selected or re-selected interconnections (colored white).

Logging Off

To log off the Command MQ On Ramp session, perform *one* of the following actions:

- Click **Logoff**.
- Close the Command MQ On Ramp frame.
- Close the Web browser.

If you do not close the Web browser, Command MQ On Ramp closes the Command MQ On Ramp window and displays its logon panel again in the Web browser.

Diagnostics

If Command MQ On Ramp is not performing correctly, one of the following situations could be causing the problem:

- The Java Virtual Machine (JVM) does not have enough storage to run the applet.
 - Look in the browser's Java console for Java error messages.
 - Try using a different vendor's JVM, such as Netscape instead of Microsoft Explorer.

- Security settings might be different.

A firewall or proxy server might prevent Java from connecting to AutoOPERATOR on OS/390. Do not access OS/390 from outside a firewall.

- The Java code is not correct.
 - Look in the browser's Java console for Java error messages.
 - Try running the Java Activator. The Activator is a standard JVM from SUN. If Command MQ On Ramp fails in the Activator, there is a Java coding problem.
- There is a communication problem with the AutoOPERATOR GME.
 - Click **Windows' Journal** to look for errors in the Command MQ On Ramp journal.
 - Click **Windows' GME** to verify a connection exists and it has no errors.
 - Verify that no errors are reported in the BBI Journal.
 - Issue the BBI commands **.D GME** and **.D GMECONN** to verify the connection is valid.
 - Turn on **DEBUGMSG=YES** in **AAOGMExx** and enter **RESET GME,RESYNC**.

If you are unable to diagnose the problem, copy the following items and fax or e-mail them to the BMC Software support team:

- Trace messages from the Java console in the browser window
- Journal messages from the Command MQ On Ramp journal
- Screen images of the GME connection details

Connecting Queue Managers Manually

If you do not use Command MQ On Ramp to configure your queue managers, you must manually define OS/390 and non-OS/390 queue managers and set up AutoOPERATOR to listen for non-OS/390 instrumentation events and issue commands to non-OS/390 queue managers.

Setting Up OS/390 Queue Manager Connectivity

MAINVIEW AutoOPERATOR for MQSeries can automate MQSeries instrumentation events from any message queue in an OS/390 queue manager. These OS/390 queue managers must be running on the same OS/390 image as MAINVIEW AutoOPERATOR for MQSeries. Command MQ On Ramp ensures that these events are enabled.

To enable AutoOPERATOR to issue commands to the local queue manager, you must ensure that the command server is started. For information about the command server and starting it, refer to the IBM manual, *IBM MQSeries Programmable System Management*.

Setting Up Non-OS/390 Queue Manager Connectivity

MAINVIEW AutoOPERATOR for MQSeries can automate events from non-OS/390 queue managers. To do this, you need to send the non-OS/390 event messages to a local queue automated by MAINVIEW AutoOPERATOR for MQSeries.

Event data is sent to the local instrumentation queues on the OS/390 MQSeries queue manager by defining the system event queues on the non-OS/390 operating system as remote queues. MQSeries automatically transmits the messages from the non-OS/390 queue manager to the OS/390 queue manager. MAINVIEW AutoOPERATOR for MQSeries, monitoring the local queues on the OS/390 queue manager, receives from the non-OS/390 queue manager, the system event messages and performs whatever automation has been defined.

To define connectivity between MAINVIEW AutoOPERATOR for MQSeries and non-OS/390 queue managers, you must

- Set up MAINVIEW AutoOPERATOR for MQSeries to listen for non-OS/390 instrumentation events. See “Setting Up MAINVIEW AutoOPERATOR for MQSeries to Listen for Non-OS/390 Instrumentation Events” on page 2-24.
- Set up MAINVIEW AutoOPERATOR for MQSeries to issue commands to non-OS/390 queue managers. See “Setting Up MAINVIEW AutoOPERATOR for MQSeries to Issue Commands to Non-OS/390 Queue Managers” on page 2-26.

Setting Up MAINVIEW AutoOPERATOR for MQSeries to Listen for Non-OS/390 Instrumentation Events

To set up AutoOPERATOR to listen for non-OS/390 instrumentation events, you must define the non-OS/390 sender and receiver channels and the transmission queue. The following is an example of how to define the channels and transmission queue when using TCP as the transmission type:

- To define the sender channel at the non-OS/390 queue manager, issue the following MQSC command:

```
DEFINE CHANNEL(nonOS/390qmgrid.to.OS/390qmgrid) +  
  CHLTYPE(SDR) +  
  CONNAME(OS/390 ip address) +  
  XMITQ(OS/390qmgrid.xmitq) +  
  CONVERT(YES)  
  TRPTYPE(TCP)
```

- To define the transmission queue on the non-OS/390 queue manager, issue the following MQSC command:

```
DEFINE QLOCAL(OS/390qmgrid.xmitq)+  
  USAGE(XMITQ)
```

- To define the receiver channel at the OS/390 queue manager, issue the following MQSC command:

```
DEFINE CHANNEL(nonOS/390qmgrid.to.OS/390qmgrid)+  
  CHLTYPE(RCVR)+  
  TRPTYPE(TCP)
```

- If PATROL is not running on the non-OS/390 operating system, issue these MQSC commands to delete the local system event queues:

```
DELETE QLOCAL (SYSTEM.ADMIN.QMGR.EVENT)+
PURGE
DELETE QLOCAL (SYSTEM.ADMIN.PERFM.EVENT)+
PURGE
DELETE QLOCAL (SYSTEM.ADMIN.CHANNEL.EVENT)+
PURGE
```

```
DEFINE QREMOTE (SYSTEM.ADMIN.QMGR.EVENT)+
  RNAME(SYSBMC.DISTRIB.EVENTS)+
  RQMNAME(OS/3900qmgrid)+
  XMITQ(OS/390qmgrid.xmitq)
DEFINE QREMOTE (SYSTEM.ADMIN.PERFM.EVENT)+
  RNAME(SYSBMC.DISTRIB.EVENTS)+
  RQMNAME(OS/390qmgrid)+
  XMITQ(OS/390qmgrid.xmitq)
DEFINE QREMOTE (SYSTEM.ADMIN.CHANNEL.EVENT)+
  RNAME(SYSBMC.DISTRIB.EVENTS)+
  RQMNAME(OS/390qmgrid)+
  XMITQ(OS/390qmgrid.xmitq)
```

- For a non-OS/390 system using a PATROL MQ product, do the following action:

Using AMQSPUT or another utility, place a message on the BMC.LISTENER.COM command queue with this content:

```
subscribe A <OS/390_qmgr> SYSTEM.DISTRIB.EVENTS
```

where <OS/390_qmgr> is the Queue Manager name from which the MAINVIEW AutoOPERATOR expects the instrumentation events.

For more information on the publish/subscribe function of ServiceView for MQ, see the *ServiceView for MQ User Guide*. For more information on defining system event queues as local queues, see the *Command MQ for D/S Installation and Administration Guide*.

Setting Up MAINVIEW AutoOPERATOR for MQSeries to Issue Commands to Non-OS/390 Queue Managers

To set up MAINVIEW AutoOPERATOR to issue commands to a non-OS/390 queue manager, you must define the sender and receiver channels, the transmission queue, and both an OS/390 and a non-OS/390 queue manager alias. The following is an example of defining the channels, transmission queue, and queue manager aliases when using TCP as the transmission type:

- To define the sender channel at the OS/390 queue manager, type

```
DEFINE CHANNEL (OS/390qmgrid.to.nonOS/390qmgrid)+
  CHLTYPE(SDR)+
  CONNAME (non-OS/390 ip address)+
  XMITQ (nonOS/390qmgrid.xmitq)+
  TRPTYPE(TCP)
```

- To define the transmission queue on the OS/390 queue manager, type

```
DEFINE QLOCAL (nonOS/390qmgrid.xmitq)+
  USAGE (XMITQ)
```

- To define the receiver channel at the non-OS/390 queue manager, type

```
DEFINE CHANNEL (OS/390qmgrid.to.nonOS/390qmgrid)+
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)
```

- To define the OS/390 queue manager alias name on the non-OS/390 queue manager, type:

```
DEFINE QREMOTE (OS/390qmgrid)+
  RQMNAME (OS/390qmgrid)+
  XMITQ (OS/390qmgrid.xmitq)
```

- To define the non-OS/390 queue manager alias name on the OS/390 queue manager, type

```
DEFINE QREMOTE (nonOS/390qmgrid)+
  RQMNAME (nonOS/390qmgrid)+
  XMITQ (nonOS/390qmgrid.xmitq)
```

Note: You must define queue manager aliases for AutoOPERATOR to issue commands to non-OS/390 queue managers and for responses to return to OS/390.

If you have not set up MAINVIEW AutoOPERATOR for MQSeries to listen for non-OS/390 instrumentation events, you must do steps 1 through 3 in “Setting Up MAINVIEW AutoOPERATOR for MQSeries to Listen for Non-OS/390 Instrumentation Events” on page 2-24.

Step 4. Using the Installation Verification Procedure

This step describes how to use the installation verification procedure. The installation verification procedure is a compiled REXX EXEC named QMQIVP00 that tests the configuration of the AutoOPERATOR environment and verifies the current status of MQSeries instrumentation event categories and AutoOPERATOR’s ability to automate events.

Before using the installation verification procedure, you must complete “Step 3. Defining Connectivity” on page 2-14. After defining the needed queues and channels, you must start the MQSeries command server and the MQSeries listener tasks. On OS/390, the command server normally starts automatically after the queue manager starts.

After the command servers and listeners are active, the sender channels on both sides of the connection can be started. For more information on starting command servers, listeners, and channels, see the publication *IBM MQSeries Command Reference*.

Use the QMQIVP00 EXEC to verify that you have defined and connected to OS/390 and non- OS/390 queue managers successfully. This EXEC can be scheduled from the BBI Log or from within an EXEC.

See Appendix A, “Diagnosing MAINVIEW AutoOPERATOR for MQSeries Errors” for a description of how to diagnose installation errors.

The syntax for invoking the QMQIVP00 EXEC from the BBI Log is

COMMAND ==> %QMQIVP00 *function* LM RM CQ

where

Function	<p>Is a required user-specified value Possible values are</p> <ul style="list-style-type: none"> • IE creates an OS/390 instrumentation event condition. Use this to test AutoOPERATOR's connection to the OS/390 queue manager and AutoOPERATOR's ability to automate events on that queue manager • CMD displays the status of OS/390 instrumentation event conditions for the specified queue manager • NCMD displays the status of OS/390 instrumentation event conditions for the specified queue manager • NIE creates a non-OS/390 instrumentation event condition. Use this to test AutoOPERATOR's connection to the non-OS/390 queue manager and AutoOPERATOR's ability to automate events on that queue manager
LM	<p>Is a required value where you type the local queue manager ID You must type a local queue manager name to use the EXEC even if you are trying to query the status of non-OS/390 queue managers.</p>
RM	<p>Is the local queue manager's alias name of a non-OS/390 queue manager. This field is required when you are querying a non-OS/390 (or remote) queue manager. The local queue manager alias name of a non-OS/390 queue manager is created in Step 5 of "Setting Up MAINVIEW AutoOPERATOR for MQSeries to Issue Commands to Non-OS/390 Queue Managers" on page 2-26. This is the name that must be used when you are querying a non-OS/390 (or remote) queue manager with this EXEC.</p>
CQ	<p>Is an optional value where you type the name for a non-OS/390 command queue. The default value is SYSTEM.ADMIN.COMMAND.QUEUE. This parameter is not applicable to OS/390 queue managers.</p>

Following are some examples about how to use the QMQIVP00 EXEC to verify that you have successfully completed implementing communications between AutoOPERATOR for MQSeries and MQSeries OS/390 and non-OS/390 queue managers.

Example

Example 1: Displaying the status in the BBI Log of OS/390 instrumentation event conditions for a local queue manager named csbe:

Command ==> %QMQIVP00 cmd csbe

The results are displayed in the BBI Log.

```
.QMQIVP00 - DISPLAY STATUS OF CURRENT INSTRUMENTATION
EVENT CATEGORI
.QMQIVP00 - CSQM409I QMNAME(CSBE)
.QMQIVP00 - INHIBTEV(ENABLED)
.QMQIVP00 - LOCALEV(ENABLED)
```

```
.QMQUIP00 - REMOTEEV(ENABLED)
.QMQUIP00 - STRSTPEV(ENABLED)

.QMQUIP00 - PERFMEV(ENABLED)
```

This output shows the name of the local queue manager and the status of the instrumentation events in that queue manager. Use this output to determine which instrumentation events are enabled.

Example

Example 2: Creating an OS/390 instrumentation event condition for a local queue manager named csbe:

```
Command ==> %QMQUIP00 ie csbe
```

The results are displayed in the BBI Log.

```
.QMQUIP00 - OS/390 INSTRUMENTATION CONDITION

.QMQUIP00 - SUCCESSFULLY CREATED
```

This output shows that both AutoOPERATOR and the OS/390 queue manager are properly configured to allow AutoOPERATOR to automate instrumentation events on the specified OS/390 queue manager.

Example

Example 3: Displaying the status of non-OS/390 (remote) instrumentation event conditions for a remote queue manager whose local queue manager alias is epesin:

Command ==> %QMQUIVP00 ncmd csbe epesin

The results are displayed in the BBI Log.

```
.QMQUIVP00 - DISPLAY STATUS OF CURRENT INSTRUMENTATION
EVENT CATEGORI
.QMQUIVP00 - AMQ8408: Display Queue Manager details.
.QMQUIVP00 - QMNAME(EPESIN)
.QMQUIVP00 - AUTHOREV(DISABLED)
.QMQUIVP00 - INHIBTEV(DISABLED)
.QMQUIVP00 - LOCALEV(DISABLED)
.QMQUIVP00 - REMOTEEV(DISABLED)
.QMQUIVP00 - PERFMEEV(DISABLED)

.QMQUIVP00 - STRSTPEV(ENABLED)
```

This output shows the name of the non-OS/390 queue manager and the status of the instrumentation events on that remote queue manager. Use this output to determine which instrumentation events are enabled.

Example

Example 4: Creating a non-OS/390 instrumentation event condition for a remote queue manager whose local queue manager alias is epesin:

Command ==> ,

The results are displayed in the BBI Log.

```
.QMQUIVP00-NON OS/390INSTRUMENTATIONEVENTCONDITION
.QMQUIVP00 - SUCCESSFULLY CREATED
```

This output shows that all of AutoOPERATOR, the OS/390 queue manager, and the remote queue manager are properly configured to allow AutoOPERATOR to automate instrumentation events on that remote queue manager.

- Step 2** Modify BBPARM member AAOMQL00 to exclude MQ System Administration queues (SYSTEM.ADMIN.????EVENT).
- Step 3** Modify BBPARM member AAOMQL00 to include SYSBMC.xxxx.EVENTS where xxxx is the four-character MAINVIEW AutoOPERATOR subsystem ID. MAINVIEW AutoOPERATOR receives local events through this queue. Ensure that SYSBMC.xxxx.EVENTS for other MAINVIEW AutoOPERATOR subsystems have been excluded, otherwise multiple copies of each event will be received.
- Step 4** Add the MQEVLPRC= parameter in BBPARM member AAOPRM00. Specify the PROC name specified for the event listener during the installation process from the PATROL for MQ - Operator version 2.2.00 or PATROL for MQ - Administrator version 3.1.00 products.
- Step 5** Remove BV from the Rule Set list specified in the AAOPRM00 member. This Rule Set is required for older releases of the PATROL products only.

MAINVIEW AutoOPERATOR for MQSeries BBI Commands

This section describes the new output for current BBI control commands that is displayed when the MAINVIEW AutoOPERATOR for MQSeries component is installed. You can use these commands on any command line when you want to see the status of the MAINVIEW AutoOPERATOR for MQSeries component or when you want to refresh the list of queues to which MAINVIEW AutoOPERATOR for MQSeries is connected.

BBI Control Command	Output Description
.DISPLAY ACTIVE or .D ACTIVE	When MAINVIEW AutoOPERATOR for MQSeries is active, issuing this command shows that the MAINVIEW AutoOPERATOR for MQSeries component is active.
.DISPLAY PRODUCTS or .D PRODUCTS	When MAINVIEW AutoOPERATOR for MQSeries is installed, issuing this command shows that the MAINVIEW AutoOPERATOR for MQSeries component is installed.
.DISPLAY KEYS or .D KEYS	When MAINVIEW AutoOPERATOR for MQSeries is installed, issuing this command shows the status of the MAINVIEW AutoOPERATOR for MQSeries component key.
.RESET MQ	Causes MAINVIEW AutoOPERATOR for MQSeries to read the currently active AAOMQLxx member and scan all the currently existing MQ queues, thereby refreshing the data regarding which MQ queues are eligible for automation.
.RESET MQ xx	Enables you to activate a different AAOMQLxx member where xx is the two-character suffix of an AAOMQLxx member.
.RESET QM nnnn	Causes MAINVIEW AutoOPERATOR for MQSeries to stop and restart the connection to a single MQSeries queue manager where nnnn is the four-character name of the queue manager. For example: .RESET QM CSQ1 The queue manager name must be specified.

Chapter 3 Customizing MAINVIEW AutoOPERATOR for MQSeries

This chapter describes parameters in BBPARM member AAOPRMxx and the BBPARM member for MQSeries, AAOMQLxx.

Using the parameters in these members you can specify which MQSeries queues will be monitored (or listened to) by MAINVIEW AutoOPERATOR and customize which actions MAINVIEW AutoOPERATOR for MQSeries will perform on your system. This chapter also describes how to enable instrumentation events.

Parameters for BBPARM Member AAOPRMxx

Table 3-1 on page 3-2 describes the parameters contained in BBPARM member AAOPRMxx for the MAINVIEW AutoOPERATOR for MQSeries component.

You can specify that each BBI-SS PAS use its own AAOPRMxx member where xx is a user-specified two-character suffix. In the UBBPARM member CFGssidA (where ssid is the subsystem ID), specify the AAOPRMxx member name for the subsystem. For more information, refer to the *MAINVIEW Common Customization Guide*.

Table 3-1 BBPARM Member AAOPRMxx Parameters

Parameter	Description
MQEV=YES NO	Specifies that MAINVIEW AutoOPERATOR for MQSeries should automatically enable instrumentation events for a queue manager during connection if it is not already enabled. The default value is NO.
MQGINHIB=xxxxxxx JRNL	Specifies the action MAINVIEW AutoOPERATOR should take when MAINVIEW AutoOPERATOR attempts to listen to a queue that is defined as GET(DISABLED). Possible settings are JRNL Issue a message to the BBI Journal stating that MAINVIEW AutoOPERATOR cannot listen to the queue. WTO Issue a write-to-operator (WTO) message stating that MAINVIEW AutoOPERATOR cannot listen to the queue. IGNORE Take no action. ALTER Alter the queue to GET(ENABLED). The default value is JRNL.
MQNSHARE=xxxxxxx JRNL	Specifies the action MAINVIEW AutoOPERATOR should take when MAINVIEW AutoOPERATOR attempts to listen to a queue that is defined as NOSHARE. Possible settings are JRNL Issue a message to the BBI Journal stating that MAINVIEW AutoOPERATOR cannot listen to the queue. WTO Issue a write-to-operator (WTO) message stating that MAINVIEW AutoOPERATOR cannot listen to the queue. IGNORE Take no action. ALTER Alter the queue to SHARE. The default value is JRNL.
MQEVLPRC=xxxxxxx	Specifies the name of the MQSeries Event Listener PROC. Only use this parameter if the MQSeries Event Listener is required. Refer to "Step 5. Setting Up MAINVIEW AutoOPERATOR for MQSeries to Co-Exist with PATROL for WebSphere MQ" on page 2-31 for more information about the MQSeries Event Listener and co-existence of MAINVIEW AutoOPERATOR and other BMC Software products that require the MQSeries event queues.

The following paragraphs describe which actions are taken, depending on the parameters you specify for MAINVIEW AutoOPERATOR for MQSeries, when MAINVIEW AutoOPERATOR tries to automate MQSeries events.

Generating Instrumentation Events

Instrumentation events, recognized by MAINVIEW AutoOPERATOR by their format, MQFMT_EVENT, are generated by queue managers. It might be possible to configure a queue manager to *not generate* instrumentation events. In this case, MAINVIEW AutoOPERATOR will not be able to hear them and, therefore, Rules for these events will not fire.

Use the AAOPRMxx parameter MQEV=YES|NO to specify whether or not MAINVIEW AutoOPERATOR for MQSeries will automatically alter queue managers to generate instrumentation events. If the queue manager has not been configured to generate instrumentation events and you choose the default setting of MQEV=NO; MAINVIEW AutoOPERATOR will not hear the events, and *MQS event Rules will not fire*.

For MAINVIEW AutoOPERATOR to hear the events, you must specify MQEV=YES. When MQEV=YES is specified, MAINVIEW AutoOPERATOR automatically alters queue managers when it is started, to generate instrumentation events *if they had been originally disabled*.

When the MQEV parameter is left with the default setting of NO (MQEV=NO), MAINVIEW AutoOPERATOR *will not* automatically alter queue managers to generate instrumentation events. Warning messages are issued during MAINVIEW AutoOPERATOR connection to a queue manager to inform you about any classes of events that are not enabled.

Encountering Queues Set to Get(Disabled)

You can specify which actions MAINVIEW AutoOPERATOR for MQSeries will take when you have set MAINVIEW AutoOPERATOR to listen to a queue that has been set to GET(DISABLED). Messages cannot be retrieved (MQGET) from a queue that has been set to GET(DISABLED). Therefore, MAINVIEW AutoOPERATOR cannot listen to any messages placed into this queue.

If you have asked MAINVIEW AutoOPERATOR to listen to a queue that is Get(Disabled), then you can use the MQGINHIB parameter in BBPARM member AAOPRMxx to take one of the following actions with these keywords:

JRNL	Issue a message to the BBI Journal stating that MAINVIEW AutoOPERATOR cannot listen to the queue.
WTO	Issue a write-to-operator (WTO) message stating that MAINVIEW AutoOPERATOR cannot listen to the queue. The message will also be issued to the BBI Journal.
IGNORE	Take no action.
ALTER	Alter the queue to GET(ENABLED).

Encountering Queues Set as NOSHARE

You can specify which actions MAINVIEW AutoOPERATOR for MQSeries takes when you set MAINVIEW AutoOPERATOR to listen to a queue that has been set to NOSHARE. If a queue has been set to NOSHARE, only one application can have the queue open. Therefore, if an application is putting messages onto a queue that is NOSHARE, MAINVIEW AutoOPERATOR will not be able to open the queue for monitoring (listening) unless some action is taken.

If you have asked MAINVIEW AutoOPERATOR to listen to a queue that is NOSHARE, you can use the MQNSHARE parameter in BBPARM member AAOPRMxx to take one of the following actions with these keywords:

JRNL	Issue a message to the BBI Journal stating that MAINVIEW AutoOPERATOR does not listen to the queue.
WTO	Issue a write-to-operator (WTO) message stating that MAINVIEW AutoOPERATOR will not listen to the queue. The message will also be issued to the BBI Journal.
IGNORE	Take no action.
ALTER	Alter the queue to share.

Note: The ALTER specification affects all queues eligible for automation as specified in AAOMQLxx.

Parameters for BBPARM Member AAOMQLxx

AAOMQLxx is a required BBPARM member used by MAINVIEW AutoOPERATOR for MQSeries. The AAOMQLxx member contains parameters that specify which MQ queues can be monitored (or listened to) by MAINVIEW AutoOPERATOR. You must have at least one valid specification for a queue in an AAOMQLxx member.

Specifying Queues

Use the BBPARM member AAOMQLxx parameters TYPE, QMGR, and QUEUE to specify MQ queues that will become eligible for automation. A fourth optional parameter, OPEN, indicates how the queue should be processed. *All parameters must be typed on one line (abbreviations are available)*; the syntax is

```
TYPE(INCL|EXCL) QMGR(queue-manager-name) QUEUE(queue-name)
      OPEN(S,I)
```

The following table describes the AAOMQLxx parameters.

Table 3-2 AAOMQLxx Parameters (Part 1 of 2)

Parameter	Description
TYPE(INCL EXCL)	Specifies whether MAINVIEW AutoOPERATOR for MQSeries should include or exclude this queue in the set of queues that are eligible for automation. This parameter is abbreviated with T. Possible values are INCL Specifies that MAINVIEW AutoOPERATOR for MQSeries should make this queue eligible for possible automation. Abbreviated with I, this setting is the default. EXCL Specifies that MAINVIEW AutoOPERATOR for MQSeries should not make this queue eligible for automation. This parameter is abbreviated with E.
QMGR(queue-manager-name)	Specifies the four-character ID for a local OS/390 queue manager that MAINVIEW AutoOPERATOR for MQSeries will monitor. Abbreviated with M, this parameter supports asterisk (*) and plus (+) wildcard characters. This is a required parameter and there is no default value. <i>You must specify a queue manager name</i> or a partial queue manager name with wildcard characters (a plus (+) represents one character, and an asterisk (*) represents one or more characters).
QUEUE(queue-name)	Specifies the 48-character name of the queue that MAINVIEW AutoOPERATOR for MQSeries makes eligible for automation. Abbreviated with U, this parameter supports asterisk (*) and plus (+) wildcard characters. This is a required parameter and there is no default value. <i>You must specify a queue name</i> or a partial queue name with wildcard characters (a plus (+) represents one character, and an asterisk (*) represents one or more characters).

Table 3-2 AAOMQLxx Parameters (Part 2 of 2)

Parameter	Description
OPEN(option1,option2)	<p>Gives you control over how the queue is opened and what action to take against the messages in the opened queue. Abbreviated with O, the subparameters are</p> <p>option 1:</p> <p>EXCLUSIVE Open queue with MQOO_INPUT_EXCLUSIVE option. This option means that other applications will be unable to open the queue while MAINVIEW AutoOPERATOR has it open. It also means that if another application has the queue opened, MAINVIEW AutoOPERATOR will be unable to open it. This parameter is abbreviated with E.</p> <p>SHARED Open queue with MQOO_INPUT_SHARED option. Abbreviated with S. This subparameter is the default for option 1.</p> <p>option 2:</p> <p>PROCESSOLD Route all existing messages found on the queue at open time through the Rule Processor so that they are automated. This option is useful for processing messages that were put on the queue while MAINVIEW AutoOPERATOR was unable to open the queue. This parameter is abbreviated with P.</p> <p>IGNOREOLD Ignore (do not process) any messages found on the queue at open time. Abbreviated with I. This subparameter is the default for option 2.</p> <p>Example 1: OPEN(EXCLUSIVE,PROCESSOLD) or O(E,P) Indicates that the queue is to be opened with the QOO_INPUT_EXCLUSIVE option and all existing messages found on the queue at open time should be rerouted through the Rule Processor for automation.</p> <p>Example 2: OPEN(SHARED,IGNOREOLD) or O(S,I) Indicates that the queue is to be opened with the MQOO_INPUT_SHARED option and all existing messages found on the queue at open time should be ignored.</p>

You can specify multiple sets of these parameters. Continuations are not supported.

Example of Specifying Queues in AAOMQLxx

BMC Software recommends that you use Figure 3-1 as a model for your specifications.

Figure 3-1 Example of Coding in AAOMQLxx Member

*PARAMETER(DEFAULT)	VALUES -----	DESCRIPTION-----
*TYPE(INCL)		TYPE ABBREVIATED WITH 'T'.
*	INCL	INCLUDE THIS QUEUE FOR RULES (INCL) OR
*		ABBREVIATE WITH 'I'.
*		EXAMPLE: T(I)
*		
*	EXCL	EXCLUDE (EXCL) THIS QUEUE FOR RULES.
*		ABBREVIATE WITH 'E'.
*		EXAMPLE: T(E)
*		
*QMGR		QMGR ABBREVIATED WITH 'M'.
*		NAME OF MQSERIES QUEUE MANAGER. NO
*		DEFAULT, THIS PARAMETER MUST BE CODED.
*		EXAMPLE: M(CSQ1)
*		
*QUEUE		QUEUE ABBREVIATED WITH 'U'.
*		NAME OF MQSERIES QUEUE. NO DEFAULT,
*		THIS PARAMETER MUST BE CODED.
*		EXAMPLE: U(SYSTEM.ADMIN.QMGR.EVENT)
*		
*OPEN(S,I)		OPEN ABBREVIATED WITH 'O'.
*	EXCLUSIVE	QUEUE OPENED WITH MQOO_INPUT_EXCLUSIVE
*		OPTION. ABBREVIATE WITH 'E'
*		
*	SHARED	QUEUE OPENED WITH MQOO_INPUT_SHARED
*		OPTION. ABBREVIATE WITH 'S'. THIS IS
*		A DEFAULT OPTION.
*		
*	PROCESSOLD	ROUTE ALL EXISTING MESSAGES FOUND ON
*		THE QUEUE AT OPEN TIME THROUGH THE RULE
*		PROCESSOR TO ALLOW AUTOMATION TO BE
*		PERFORMED ON THEM. ABBREVIATE WITH 'P'.
*		
*	IGNOREOLD	IGNORE (DO NOT PROCESS) ANY EXISTING
*		MESSAGES FOUND ON THE QUEUE AT OPEN TIME.
*		ABBREVIATE WITH 'I'. THIS IS A DEFAULT
*		OPTION.
*		
*		EXAMPLE 1: OPEN(EXCLUSIVE,PROCESSOLD) OR
*		O(E,P) INDICATES THE QUEUE IS TO BE
*		OPENED WITH THE MQOO_INPUT_EXCLUSIVE
*		OPTION AND ALL EXISTING MESSAGES FOUND ON
*		THE QUEUE AT OPEN TIME SHOULD BE ROUTED
*		THROUGH THE RULE PROCESSOR FOR
*		AUTOMATION.
*		
*		EXAMPLE 2: OPEN(SHARED,IGNOREOLD) OR
*		O(S,I) INDICATES THE QUEUE IS TO BE
*		OPENED WITH THE MQOO_INPUT_SHARED OPTION
*		AND ALL EXISTING MESSAGES FOUND ON THE
*		QUEUE AT OPEN TIME SHOULD BE IGNORED.
*		

```

*-----
* MQSERIES SYSTEM EVENT QUEUES
*-----
TYPE(INCL) QMGR(*) QUEUE(SYSTEM.ADMIN.*.EVENT)
*
*-----
* MQSERIES SYSTEM EVENT'S FROM NON-OS/390 QUEUE MANAGERS
*-----
TYPE(INCL) QMGR(*) QUEUE(SYSTEM.DISTRIB.EVENTS)
* REMAINDER OF QUEUES *DO NOT OPEN*
*-----
TYPE(EXCL) QMGR(*) QUEUE(*)

```

Note: In the list, you must separately list the queues eligible for automation from the queues that will be excluded from automation.

For each MQSeries queue manager, MAINVIEW AutoOPERATOR matches the list of existing queues to the contents of the active AAOMQLxx member. It sequentially scans the AAOMQLxx member and evaluates the QMGR(queue manager name) and QUEUE(queue name) specifications until a match is found.

When a match is found MAINVIEW AutoOPERATOR will either include or exclude the queue based on the TYPE(INCL|EXCL) setting. If two (or more) statements overlap or conflict, the first definition encountered will be used.

Any queue not included in the AAOMQLxx member will not be eligible for automation. If you code a Rule for an excluded queue, the Rule will never fire.

The AAOMQLxx is processed

- at BBI-SS PAS startup
- when the BBI commands .RESET MQ or .RESET QM are issued

For more information about BBI commands for MAINVIEW AutoOPERATOR for MQSeries, refer to “MAINVIEW AutoOPERATOR for MQSeries BBI Commands” on page 2-33.

Note: While MAINVIEW AutoOPERATOR has a queue opened, no other application can open exclusively or delete that queue.

Chapter 4 Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics

This chapter describes

- how to view MAINVIEW AutoOPERATOR for MQSeries automation statistics from the MAINVIEW AutoOPERATOR for MQSeries Workstation
- how to use BBI control commands for MAINVIEW AutoOPERATOR for MQSeries

Using the MAINVIEW AutoOPERATOR for MQSeries Workstation

MAINVIEW AutoOPERATOR for MQSeries provides the MQSeries Workstation panel (Figure 4-1). You can use this panel to view both performance and queue information from a single panel. As with all other BBI displays, the MQSeries Workstation can be pointed to from one BBI-SS PAS target to another.

Use the MQSeries Workstation panel, shown in Figure 4-1, to view statistics that show how much automation of MQSeries events has been accomplished and to see which queues are active for this BBI-SS PAS.

Figure 4-1 MQSeries Workstation Panel

```

BMC Software ----- MQSeries Workstation ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> EP01
Interval ==> 3      INPUT                         DATE --- 02/02/11
Commands: ALL, EQI xxx, NEQI xxx, EE xxx, NEE xxx    TIME --- 08:20:32
----- Performance Statistics -----
Total Queues Included                               6
Total MQS Messages Arrived                          1   Total MQS Messages Handled          0
Instrument. Events Arrived                          1   Instrument. Events Handled          0
User Messages Arrived                              0   User Messages Handled              0
Current Instr.Arrival Rate/Min                      0   Peak Instr. Arrival Rate/Min       0
Rule Generated Alerts                               0   Rule Triggered EXECs               0
----- Queue Management -----
Queue Manager           Queues   Event Ques   Enabled
                        Incl./Total   Included     Events
-----
CSQ1                    6/24                               SLIPR
***** Bottom of Data *****

```

This panel is divided into two parts:

- Performance Statistics
- Queue Management

With these two areas, you can see statistics on both the performance of MQSeries and the status of individual queues.

Primary Commands

Table 4-1 lists and describes all of the primary commands that you can type on the MQSeries Workstation panel.

Table 4-1 MQSeries Workstation Panel Primary Commands (Part 1 of 2)

Primary Command	Description
ALL	Shows all MQSeries queue managers for which MAINVIEW AutoOPERATOR for MQSeries will attempt automation.
eqi xxx	Shows only queue managers where MAINVIEW AutoOPERATOR for MQSeries exists, including instrumentation events for automation. Use xxx to specify which instrumentation events you want to view, where xxx may be any (or all) of the following values: Q system.admin.qmgr.event P system.admin.perfm.event C system.admin.channel.event For example, to view only the queue managers that include system.admin.qmgr.event and system.admin.channel.event, type EQI QC This command cannot be used if MAINVIEW AutoOPERATOR coexists with the PATROL for WebSphere MQ.
neqi xxx	This command does the opposite of the EQI command. With this command, you can specify which queues not to view. Use xxx to specify which instrumentation events you do not want to view, where xxx may be any (or all) of the following values: Q system.admin.qmgr.event P system.admin.perfm.event C system.admin.channel.event For example, to view the queue managers that do not include SYSTEM.ADMIN.CHANNEL.EVENT, type NEQI C This command cannot be used if MAINVIEW AutoOPERATOR coexists with PATROL for WebSphere MQ.
ee xxx	Shows only queue managers with xxx events enabled, where xxx may be any (or all) of the following values: S STRSTPEV (Start/Stop event) L LOCALEV (Local event) I INHIBTEV (Inhibit event) P PERFMEV (Performance event) R REMOTEV (Remote event) For example, to view only those queue managers which include remote events, type EE R

Table 4-1 MQSeries Workstation Panel Primary Commands (Part 2 of 2)

Primary Command	Description
nee xxx	<p>This command works the opposite of the EE xxx. Use this command to show only the queue managers that do not have xxx events enabled, where xxx may be any one (or all) of the following:</p> <ul style="list-style-type: none"> s strstpev (Start/Stop event) l localev (Local event) i inhibtev (Inhibit event) p perfmev (Performance event) r remotev (Remote event) <p>For example, to view the queue managers which do not include remote events, type: nee r</p>

Note: The status of events is taken when MAINVIEW AutoOPERATOR for MQSeries connects to the queue manager. If the status is changed, MAINVIEW AutoOPERATOR for MQSeries must be disconnected and then reconnected to obtain the correct status. To request this status change, use the BBI control command **.RESET:**

```
.RESET QM xxxx
```

where xxxx is the four-character queue manager ID.

MQSeries Workstation Fields

Figure 4-2 shows the fields of the MQSeries Workstation panel.

Figure 4-2 MQSeries Workstation Panel Fields

```

BMC Software ----- MQSeries Workstation ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> EP01
Interval ==> 3      INPUT                          DATE --- 02/02/11
Commands: ALL, EQI xxx, NEQI xxx, EE xxx, NEE xxx    TIME --- 08:20:32
----- Performance Statistics -----
Total Queues Included                6
Total MQS Messages Arrived           1  Total MQS Messages Handled           0
Instrument. Events Arrived           1  Instrument. Events Handled           0
User Messages Arrived                0  User Messages Handled               0
Current Instr.Arrival Rate/Min       0  Peak Instr. Arrival Rate/Min       0
Rule Generated Alerts                0  Rule Triggered EXECs               0
----- Queue Management -----
                               Queues   Event Ques   Enabled
Queue Manager                   Incl./Total  Included     Events
-----
CSQ1                             6/24                               SLIPR
***** Bottom of Data *****

```

Performance Statistics

Figure 4-3 shows the Performance Statistics portion of the MQSeries Workstation panel.

Figure 4-3 MQSeries Workstation - Performance Statistics

```

----- Performance Statistics -----
Total Queues                174  Total Queues Included           9
Total MQS Messages Included  1  Total MQS Messages Handled      1
Instrument. Events Included  1  Instrument. Events Handled       1
User Messages Included       0  User Messages Handled            0
Current Instr.Arrival Rate/Min 0  Peak Instr. Arrival Rate/Min    0
Rule Generated Alerts        0  Rule Triggered EXECs            0

```

Use this portion of the panel when you are trying to evaluate how often the automation of MQSeries events is occurring on your system.

The **Performance Statistics** field names are described in Table 4-2.

Table 4-2 Performance Statistics Field Names

Field Name	Description
Total Queues	total number of queues that are found in all currently connected queue managers
Total Queues Included	total number of queues that are eligible for automation by MAINVIEW AutoOPERATOR for MQSeries
Total MQS Messages Included	total number of MQS messages that are seen by MAINVIEW AutoOPERATOR (eligible for automation)
Total MQS Messages Handled	total number of MQ messages where at least one Rule has fired
Instrument. Events Included	total number of messages that are seen by MAINVIEW AutoOPERATOR (eligible for automation) that were instrumentation events
Instrument. Events Handled	total number of MQ messages that had at least one Rule fired and that were instrumentation events
User Messages Included	total number of user MQ messages that are seen by MAINVIEW AutoOPERATOR that were eligible for automation
User Messages Handled	total number of user MQ messages where at least one Rule fired
Current Instr. Arrival Rate/Min	arrival rate of instrumentation events per minute
Peak Instr. Arrival Rate/Min	highest arrival rate of instrumentation events per minute
Rule Generated Alerts	total number of event type (MQS) ALERTs that were generated by Rules
Rule Triggered EXECs	total number of event type (MQS) EXECs that were triggered by Rules

In general, the figures shown in the Performance Statistics portion of the MQSeries Workstation panel are useful after you have determined the average amount of automation activity. Then, if there is a change in statistics, you can have a better sense of whether the change is due to an abnormal condition or if it is an expected and acceptable amount.

Viewing Queue Management

Figure 4-4 shows the Queue Management portion of the MQSeries Workstation panel.

Figure 4-4 Queue Management Portion of the MQSeries Workstation Panel

```

----- Queue Management -----
Queue Manager                Queues    Event Queues  Enabled
                             Incl./Total  Included      Events
-----
CSQ3                          19/194
CSQ4                          9/174        QCP          SLIPR
***** Bottom of Data *****

```

Use this section of the panel to see which queue managers are active, the number of queues that are eligible for automation, and the total number of queues on the queue manager. This section of the panel also displays the type of Event Queues that are included for automation and the type of Events that are enabled.

The Queue Management field names are described in Table 4-3.

Table 4-3 Queue Management Field Names

Field Name	Description
Queue Manager	SSID name of the MQSeries queue manager The list includes only queue managers to which MAINVIEW AutoOPERATOR has attempted to connect. MAINVIEW AutoOPERATOR will attempt to connect to queue managers if they are defined in a Rule.
Queues Incl./Total	number of queues that MAINVIEW AutoOPERATOR is listening to for automation, followed by the total number of queues on that queue manager
Event Queues Included	instrumentation queues that are included by MAINVIEW AutoOPERATOR for automation P: Performance, Q: Queue Manager, C: Channel, ALL: All
Enabled Events	events that were found to be enabled when the connection was made to this queue manager I: Inhibit, L: Local, P: Performance, R: Remote, S: Strstp,

Chapter 5 Automating MQSeries Events

This chapter describes

- how MAINVIEW AutoOPERATOR for MQSeries automates MQSeries events with Rules
- two procedures that show how to create Rules for MQSeries events
- variables that are available to support the MQS event type
- panel and field definitions for the Rule Processor Selection Criteria and Action Specification panels for the MQS event type
- the event type MQS in the Automation Reporter Rules record

For more information about creating Rules and Rules Sets, see the “Implementing AutoRULE” chapter in the *MAINVIEW AutoOPERATOR Customization Guide*.

How MAINVIEW AutoOPERATOR for MQSeries Automates MQSeries Events

MAINVIEW AutoOPERATOR uses Rules to perform automation for various system events such as OS/390 messages, commands, and ALERTs. Rules can also be used to provide automated responses to MQSeries events. This section describes important concepts about MAINVIEW AutoOPERATOR Rules and MQSeries events.

What a Rule Is

A Rule is a two-part statement: when the conditions of the first part of the statement are met (the selection criteria), the automation actions specified in the second part are performed.

When MAINVIEW AutoOPERATOR is installed, you can create Rules that provide automated actions such as issuing commands, creating ALERTs, or rerouting messages in response to OS/390 system events. MAINVIEW AutoOPERATOR Rules listen for these events. When the event occurs, the Rule fires and the automation action is taken.

When MAINVIEW AutoOPERATOR for MQSeries is installed, you can create Rules for MQSeries events. The MQSeries events can originate either from queues within OS/390 or from non-OS/390 platforms. MAINVIEW AutoOPERATOR listens for MQSeries messages and recognizes them as event type MQS. Enabled Rules listen for these MQS messages and when the conditions of the Rule are met, the Rule fires to provide automated responses to the message.

Note: For MAINVIEW AutoOPERATOR for MQSeries to listen for and automate events from non-OS/390 queue managers (such as those running on Windows NT, UNIX, OS/2, and OS/400 operating systems), you must define connectivity and activate communications between OS/390 and non-OS/390 queue managers as described in “Step 3. Defining Connectivity” on page 2-14.

What MQS Events Are

The MAINVIEW Rule Processor listens for MQSeries messages and recognizes them as event type MQS. These messages (or events) can originate from an OS/390 or non-OS/390 queue manager. There are additional customization steps required for listening to MQS events from non-OS/390 queue managers.

MAINVIEW AutoOPERATOR distinguishes between different MQSeries messages by their format in the message descriptor. Table 5-1 on page 5-3 lists these message formats and description.

Table 5-1 Message Format and Description

Message Format	Description
MQFMT_NONE	No format
MQFMT_ADMIN	Command server request/reply message
MQFMT_CHANNEL_COMPLETED	Channel has ended
MQFMT_CICS	CICS Information Message
MQFMT_COMMAND_1	Type 1 command reply message
MQFMT_COMMAND_2	Type 2 command reply message
MQFMT_DEAD_LETTER_HEADER	Dead Letter Message
MQFMT_EVENT	Event Message
MQFMT_IMS	IMS Information Message
MQFMT_IMS_VAR_STRING	IMS Variable String Message
MQFMT_MD_EXTENSION	Message Descriptor Extension
MQFMT_PCF	Programmable Command Format
MQFMT_REF_MSG_HEADER	Reference Message Header
MQFMT_STRING	Character String
MQFMT_TRIGGER	Trigger Message
MQFMT_WORK_INFO_HEADER	Work Information Message
MQFMT_XMIT_Q_HEADER	Transmission Queue Message
MQFMT_RF_HEADER	Rules and formatting header
MQFMT_RF_HEADER_2	Rules and formatting header version 2

The origin of the message may be from an OS/390 queue manager or a non-OS/390 queue manager. When you create a Rule, you must specify the format of the message. A single Rule can recognize only one specified format. However, by specifying USER, the Rule can fire for any format except MQFMT_EVENT.

What Event Messages Are

MQSeries event messages (or instrumentation events) are provided by MQSeries and enable MAINVIEW AutoOPERATOR, through Rules, to monitor queue manager conditions, performance conditions, and channel conditions. MQSeries events originate from the three administration queues; system.admin.qmgr.event queue, the system.admin.perfm.event queue, and the system.admin.channel.event queue. However, MAINVIEW AutoOPERATOR is not restricted to receiving those messages from those queues only.

You can create a Rule to listen for MQSeries events (format MQFMT_EVENT) and subsequently take automation action. The automation action the Rule performs could be to create an ALERT to warn the operator or issue a command to copy the data in the queue to a data set.

What Non-Event Messages Are

Non-Event messages are any MQSeries messages (except messages with format MQFMT_EVENT) that arrive on a queue. For example, you could have an accounts payable (AP) application that sends data to and from MQSeries user queues. You can create Rules to fire based on the data from the Message Buffer or the various MQSeries structures that make up the message.

Rule Processor MQSeries Variables

MAINVIEW AutoOPERATOR provides special variables for MQSeries Rules. Using these variables, the Rule can fire based on any of many possible values that are within a MQSeries message, whether it is an instrumentation event or an application message. These variables are created from MQSeries structures and data in the message buffer, and can be used on panels that make up the Rule Processor user interface. These variables can be used to determine if the Rule should fire and, if so, what resources, if any, should be acted upon by the Rule. The following list describes general properties for variables available in MQSeries Rules:

Variable names consist of a

- prefix name (for example, IMFQ)
- structure name (for example, MD, DLH)
- field name (for example, MsgId or CorrelId in the message descriptor)

Each node of the variable name is delimited by an underscore (_)

Some field-name variables have associated constant names defined for them by MQSeries. In these cases, the variables contain the constant name rather than the value within the data structure. For example: in the case of a Dead Letter message, the variable IMFQ_MD_FORMAT, which contains the format field from the Message Descriptor, is set to MQFMT_DEAD_LETTER_HEADER. Likewise, the variable IMFQ_MD_VERSION, which contains the version of the message descriptor, is set to MQMD_VERSION_1 instead of x'1'.

The following rules apply for variables set by MAINVIEW AutoOPERATOR for MQSeries:

- If the value for a character field does not have a defined constant value, the field value is used.
- If the value for a numeric field that has a defined range does not have a defined constant, the numeric value converted to zoned decimal is used.
- Variables for fields that are defined as hexadecimal fields (MQBYTExx in MQSeries) are not converted and are set 'as is.'
- Variables that are created for flag fields in a MQSeries structure may contain multiple constant values, delimited by blanks. For example, the variable for the Message Descriptor Report field, IMFQ_MD_REPORT, can have several values. The resulting value might look like this: 'MQRO_COA MQRO_COD'. Any other valid combination could be seen as well.
- Variables that are created for MQSeries fields where the value has more than one defined constant will contain both constants. For example, the variable IMFQ_MD_PUTAPPLTYPE, which contains the PutApplType value from the message descriptor, would contain the value 'MQAT_OS390 MQAT_MVS' if the message originated on an OS/390 system. This result is because both constants contain the value '2'.

Enabling MQS Events

Each MQS event falls into a category of events that must be enabled within the queue manager before MAINVIEW AutoOPERATOR Rules can *hear* them.

Refer to the *IBM MQSeries Command Reference* for information about enabling these categories. During MAINVIEW AutoOPERATOR for MQSeries initialization, warning messages can be issued for each event category that is not enabled, or MAINVIEW AutoOPERATOR can enable them automatically.

To enable MQS events so that MAINVIEW AutoOPERATOR Rules can hear them, ensure that the BBPARM member AAOMQLxx lists the names of all the queues for which you want to hear events. The event queues must also be defined with the appropriate attributes as described in Chapter 3, "Customizing MAINVIEW AutoOPERATOR for MQSeries."

For information about listing the queue names in AAOMQLxx, refer to “Specifying Queues” on page 3-5.

Note: If you use the default AAOMQLxx member, MAINVIEW AutoOPERATOR listens for only MQFMT_EVENT messages that originate from administration queues such as the SYSTEM.ADMIN.QMGR.EVENT queue, the SYSTEM.ADMIN.PERFM.EVENT queue, and the SYSTEM.ADMIN.CHANNEL.EVENT queue. To listen for MQFMT_EVENT and User messages on any other queue, you must modify the AAOMQLxx member of BBPARM and include the names (or name pattern) of the queues to be monitored.

Creating Rules for Event Messages and Non-Event Messages

This section shows how to use the Rule Processor application panels to create Rules for MQSeries event messages (format MQFMT_EVENT) or non-event messages (format other than MQFMT_EVENT). For more information about Rules, other events that you can create Rules for, and examples, refer to the *MAINVIEW AutoOPERATOR Basic Automation Guide*.

Creating a Rule for a MQFMT_EVENT Message

This section shows you how to create a MAINVIEW AutoOPERATOR Rule that will listen and fire for a MQSeries event that occurs when a specific queue belonging to the queue manager (named CSQ1 in this example) becomes full. When the queue becomes full, the queue manager issues an instrumentation event (format MQFMT_EVENT) with a type of Q_FULL.

By following the example, you will learn how to create a Rule to listen for this event and when it fires, the Rule will update a variable that indicates what happened.

To create the Rule, you must enter the correct Selection Criteria information. The following table describes the information you need to enter:

You Need to Know	Which Is	To Enter It on This Panel
The Rule event type	MQS	Rule Processor Detail Control
The queue manager name	CSQ1, in this example	Selection Criteria - MQS
The message format	MQFMT_EVENT	Selection Criteria - MQS
The MQSeries event type	q_full	Selection Criteria - MQS
The name of the queue for which the event occurred	PROD.AP	Variable Dependencies

For more information about creating Rules and Rules Sets, see the “Implementing AutoRULE” chapter in the *MAINVIEW AutoOPERATOR Customization Guide*.

Example: Creating an MQFMT_EVENT Rule

To create a Rule that fires when a Q_FULL event occurs for the PROD.AP queue, follow these steps:

- Step 1** From the Automation Control panel, type the E line command, for (E)nable, and press **Enter** to enable the Rule Set to which you want to add the Rule. You can add Rules only to enabled Rule Sets; see Figure 5-1.

Figure 5-1 Creating an MQFMT_EVENT Rule: Automation Control Panel (Example 1)

```

BMC Software ----- Automation Control ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> SYSE
Primary commands: Add, Statshow, Cmdshow          DATE --- 00/11/15
                                                TIME --- 15:56:01

Automation Status      ==> ACTIVE                (Active, Inactive)
Automation Strategy    ==> INDIVIDUAL            (Individual, All, First, Qualified)
Honor MPF Suppression ==> YES                   (NO/YES)
Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (SA)ve
              (M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC  Rule-Set Status  Rules    Fired  Filtered  Date    Time    Strategy
___ AAORULM1 DISABLED N/A     N/A     N/A     N/A     N/A     N/A
___ AAORULQ2 DISABLED N/A     N/A     N/A     N/A     N/A     N/A
___ AAORULXX DISABLED N/A     N/A     N/A     N/A     N/A     N/A
___ AAORUL00 DISABLED N/A     N/A     N/A     N/A     N/A     N/A
___ AAORUL01 DISABLED N/A     N/A     N/A     N/A     N/A     N/A
___ AAORUL02 DISABLED N/A     N/A     N/A     N/A     N/A     N/A
___ RULBCPSM DISABLED N/A     N/A     N/A     N/A     N/A     N/A
e___ RULMQS01 DISABLED N/A     N/A     N/A     N/A     N/A     N/A
***** END OF DATA *****
    
```

For more information about creating Rules and Rules Sets, see the “Implementing AutoRULE” chapter in the *MAINVIEW AutoOPERATOR Customization Guide*.

Step 2 To select the Rule Set, type the **S** line command, for (S)elect, in the **LC** field next to the Rule Set name:

```

BMC Software ----- Automation Control ----- RULMQS01 ENABLED
COMMAND ==> TGT ==> SYSE
Primary commands: Add, Statshow, Cmdshow DATE --- 00/11/15
TIME --- 15:56:14

Automation Status ==> ACTIVE (Active, Inactive)
Automation Strategy ==> INDIVIDUAL (Individual, All, First, Qualified)
Honor MPF Suppression ==> YES (NO/YES)

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (SA)ve
(M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC Rule-Set Status Rules Fired Filtered Date Time Strategy
___ AAORULMQ ENABLED 83 0 72,180 15-NOV-00 11:50:08 ALL
s___ RULMQS01 ENABLED 79 0 1 15-NOV-00 15:56:10 ALL
___ AAORULBA DISABLED N/A N/A N/A N/A N/A
___ AAORULBB DISABLED N/A N/A N/A N/A N/A
___ AAORULBC DISABLED N/A N/A N/A N/A N/A
___ AAORULBD DISABLED N/A N/A N/A N/A N/A
___ AAORULBE DISABLED N/A N/A N/A N/A N/A
___ AAORULBF DISABLED N/A N/A N/A N/A N/A
___ AAORULBG DISABLED N/A N/A N/A N/A N/A
___ AAORULBH DISABLED N/A N/A N/A N/A N/A
___ AAORULBP DISABLED N/A N/A N/A N/A N/A
___ AAORULBQ DISABLED N/A N/A N/A N/A N/A
___ AAORULBR DISABLED N/A N/A N/A N/A N/A
    
```

The Rule Set Overview panel is displayed. Figure 5-2 on page 5-10 shows an example of a Rule Set named RULMQS01 and the Rules it contains.

Step 3 To add a new Rule, use the **ADD** primary command on the **COMMAND** line.

Figure 5-2 Creating an MQFMT_EVENT Rule: Rule Set Overview Panel (Example1)

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==> ADD TGT --- SYSE
Rule Set ID: RULMQS01 Ruleset Strategy ==> ALL DATE --- 00/11/15
Primary commands: Add, Save, Sort, Unsort, Reset, Filter TIME --- 15:56:28
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
(C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)repeat
Sort Criterion: right/left

LC Rule-id Stat Text-id Type Fired EXEC Changed ID
---
___ MQQDPHI2 DIS Q_DEPTH_HIGH MQS 0 MQVARS 2 97/01/24 20:30 JDB3
___ MQQDPLOW ENA Q_DEPTH_LOW MQS 0 MQVARS 1 00/11/15 15:56 BAOJDB4
___ MQQDPLO2 ENA Q_DEPTH_LOW MQS 0 MQVARS 2 00/11/15 15:56 BAOJDB4
___ MQSVINTH ENA Q_SERVICE_INTERV MQS 0 MQVARS 1 00/11/15 15:56 BAOJDB4
___ MQSVINT2 ENA Q_SERVICE_INTERV MQS 0 MQVARS 2 00/11/15 15:56 BAOJDB4
___ MQSVINOK ENA Q_SERVICE_INTERV MQS 0 MQVARS 1 00/11/15 15:56 BAOJDB4
___ MQSVINO2 ENA Q_SERVICE_INTERV MQS 0 MQVARS 2 00/11/15 15:56 BAOJDB4
___ MQPUTENA ENA PUT_INHIBITED MQS 0 MQVER MQ 97/02/19 14:51 JDB3
___ MQPUTEN2 SUS PUT_INHIBITED MQS 0 MQVER MQ 97/02/19 14:51 JDB3
___ MQQFULL ENA Q_FULL MQS 0 MQVER MQ 97/02/19 20:49 JDB3
___ QUEUE1 ENA MQS 0 VARDISP 00/11/15 15:56 BAOJDB4
***** END OF DATA *****
    
```

The Rule Processor Detail Control panel is displayed:

Figure 5-3 Creating a MQFMT_EVENT Rule: Rule Processor Detail Control Panel (Example 1)

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==> TGT --- SYSE
TIME --- 15:57:17
DATE --- 00/11/15

The following options are displayed in sequence, or may
be selected by entering the two-character code

S1 - Selection Criteria A1 - Action Specification
SV - Variable Dependencies AA - Alert Action(s) I
AD - Alert Action(s) II

Rule ID ==>
Event Type ==> Type of event ( ? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched ==> (Maximum # times matched within INTERVAL, 1-100)
in seconds ==> (Interval length, 1-99999 seconds)
then status ==> (SUSPEND, DISABLE)

Application information:
Group ==> Function ==> Code ==>
Author ==> BMCUSER Description ==>
Last Modified by on at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes
    
```

Step 4 Type the following information:

- a unique name in the Rule ID field (for example, APQHI); see Figure 5-4
- **MQS** in the **Event Type** field; see Figure 5-4

Note: The Event Type MQS will be accepted *only* if the MAINVIEW AutoOPERATOR for MQSeries key is present. If the key is not present, you receive a short message in the upper right corner of the panel.

Figure 5-4 Creating a MQFMT_EVENT Rule: Rule Processor Detail Control Panel (Example 2)

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>
TGT --- SYSE
TIME --- 15:58:13
DATE --- 00/11/15

The following options are displayed in sequence, or may
be selected by entering the two-character code

S1 - Selection Criteria           A1 - Action Specification
SV - Variable Dependencies       AA - Alert Action(s) I
                                AD - Alert Action(s) II

Rule ID      ==> APQHI
Event Type   ==> MQS      Type of event ( ? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>          (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>          (Interval length, 1-99999 seconds)
then status  ==>          (SUSPEND, DISABLE)

Application information:
Group        ==>          Function      ==>          Code      ==>
Author       ==> BMCUSER  Description ==>
Last Modified by      on          at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes
    
```

Step 5 Press Enter.

The Selection Criteria - MQS panel is displayed.

Figure 5-5 **Creating a MQFMT_EVENT Rule: Selection Criteria - MQS Panel (Example 1)**

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE

          Rule-set === RULMQS01           Rule-id  === APQHI

Queue Identification:                                (1 to 12 Queue Managers)
Manager(s)  ==>
Queue Id    ==>

Message Identification:
Format      ==>                                (Value from MD Format field)
Event Type  ==>                                (Enter ? for help)

          Sub  Len  Op  Value
Msgid      ==> ___ : ___ ___ _____
CorrelId   ==> ___ : ___ ___ _____
Msg Buffer  ==> ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 6 To enter the selection criteria for this event

- 6.A** Enter your queue manager name in the **Manager(s)** field. In this example, CSQ1 is the queue manager name; see Figure 5-6.
- 6.B** In the **Format** field, type **MQFMT_EVENT**.

This is the MQSeries message format. When you create Rules for messages with format MQFMT_EVENT, you can also use the **Event Type** field to specify the type of MQSeries event for which you can listen. This is optional. You cannot use the **MsgId**, **CorrelId** or **Msg Buffer** fields for messages of format MQFMT_EVENT.

- 6.C** For a selectable list of MQ instrumentation Events, type **?**.

Figure 5-6 Selection Criteria - MQS Panel (Example 2)

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

          Rule-set === RULMQS01                Rule-id === APQHI

Queue Identification:                               (1 to 12 Queue Managers)
Manager(s)  ==> CSQ1
Queue Id    ==>

Message Identification:
Format      ==> MQFMT_EVENT                    (Value from MD Format field)
Event Type  ==> ?                             (Enter ? for help)

          Sub  Len  Op  Value
Msgid      ==> ___ : ___ ___ _____
CorrelId   ==> ___ : ___ ___ _____
Msg Buffer  ==> ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

The MQSeries Event Types panel is displayed, which shows a scrollable list of (MQS) Event Types. See Figure 5-7.

Figure 5-7 Example of a MQSeries Event Types Panel

```

BMC Software ----- MQSeries Event Types ----- AutoOPERATOR
COMMAND ==>
Line commands: (S)elect, (B)rowse
LC Queue   Type
___ QMGR   Alias_base_Q_type_error
___ QMGR   Get_inhibited
___ QMGR   Put_inhibited
___ QMGR   Q_type_error
___ QMGR   Remote_Q_name_error
___ QMGR   Xmit_Q_type_error
___ QMGR   Xmit_Q_usage_error
___ QMGR   Unknown_alias_base_Q
___ QMGR   Unknown_object_name
___ QMGR   Unknown_remote_Q_mgr
___ Channel Channel_activated
___ Channel Channel_not_activated
___ Channel Channel_started
___ Channel Channel_stopped
B__ Perform Q_depth_high
___ Perform Q_depth_low
___ Perform Q_full
___ Perform Q_service_interval_high
___ Perform Q_service_interval_ok
    
```

Note: You can type the **B** (for Browse) for more than one Type. The additional selected panels will be displayed one at a time. To proceed from one panel to the next, press **PF3**.

Step 7 To browse descriptive information about a specific Type, type **B** in the LC (line command) column.

A Help panel is displayed. See Figure 5-8.

Figure 5-8 Example of a Help Panel for Q_DEPTH_HIGH

```

BMC Software ----- MQSeries Event Types ----- AutoOPERATOR
COMMAND ===>

Type          Q_DEPTH_HIGH

Description   A MQPUT or MQPUT1 call caused the queue depth to be incremented
              to or above the queue depth high limit.

Variables     IMFQ_EVENT_QMGRNAME      : Name of Queue Manager that generated
              the event, whether local or remote.

              IMFQ_EVENT_QNAME      : Name of user or system queue that
              caused the event.

              IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the
              statistics were last reset.

              IMFQ_EVENT_HIGHQDEPTH  : Max number of message on the queue
              since the statistics were last reset.

              IMFQ_EVENT_MSGENQCOUNT : Number of messages enqueued since
              statistics were last reset.

              IMFQ_EVENT_MSGDEQCOUNT : Number of messages dequeued since
              statistics were last reset.

QMGR Attribute      : PERFMEV
    
```

Step 8 To return to the previous panel, press **PF3**.

The MQSeries Event Types panel is displayed again (see Figure 5-7 on page 5-13).

Step 9 To select an Event Type, type **S** in the LC column next to the Event Type on the MQSeries Event Types panel and press **Enter**.

The Selection Criteria panel is displayed with the Event Type entered. See Figure 5-9 on page 5-15.

Figure 5-9 Selection Criteria - MQS Panel (Example 3)

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

          Rule-set === RULMQS01                Rule-id === APQHI

Queue Identification:                               (1 to 12 Queue Managers)
Manager(s)  ==> CSQ1
Queue Id    ==>

Message Identification:
Format      ==> MQFMT_EVENT                      (Value from MD Format field)
Event Type  ==> Q_DEPTH_HIGH                     (Enter ? for help)

          Sub  Len  Op  Value
Msgid      ==> ___ : ___ ___ _____
CorrelId   ==> ___ : ___ ___ _____
Msg Buffer  ==> ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 10 Press Enter.

The Variable Dependencies - MQS panel is displayed (see Figure 5-10 on page 5-16).

For this example, the Rule should fire only when the event occurs for the PROD.AP queue. To set up this automation, on the Variable Dependencies panel, specify that the value of the variable &IMFQ_EVENT_QNAME matches the queue name PROD.AP.

Step 12 Use the Action Specification - MQS panel to specify the action for the Rule to take when it fires. For this example, the action taken is to set the value of the STAT.PROD.AP variable to **FULL** and retain the message in the queue.

To specify the action for this Rule

- 12.A** In the **Set Variable** fields, type **STAT. &IMFQ_EVENT_QNAME** and **FULL**.
- 12.B** Verify that **YES** is in the **Keep Message** field, which tells MAINVIEW AutoOPERATOR to leave the instrumentation event message in the event queue.

Figure 5-12 Action Specification - MQS Panel (Example 2)

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==>                                                    TGT --- SYSE

          Rule-set === RULMQS01          Rule-id  === APQHI

Automation Actions:
Journal          ==>
EXEC Name/Parms ==>
Cmd (Type  MQ  ) ==>

Set Variable     ==> STAT.&IMFQ_EVENT_QNAME          ==> FULL
Notify          ==>                                Outboard Pager ID
Info            ==>

DOM Id          ==>                                Delete Operator Message
Issue WTO Msg   ==>

MQSeries Automation Actions:
Keep Message     ==> YES   (Yes or NO)   Remove DLH ==>          (Yes or No)
Destination Que  ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 13 Press **Enter**.

The Alert Action(s) I -MQS panel is displayed, as shown in the next figure.

```

BMC Software ----- Alert Action(s) I - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE
          Rule-set === RULMQS01                 Rule-id === APQHI
Function ==>                                     (ADD, DELETE, DELETEQ)
Key      ==>
Text     ==>
Queues   ==>                                     Alert Queue Name(s)
Priority ==>                                     (CRIT,MAJ,MIN,WARN,INFO,CLEAR)
Color    ==>                                     RED,PINK,YEL,DKBL,LTBL,GRE,WHI
PCMD     ==>
System   ==>                                     Return to target after PCMD
EXEC     ==>                                     Follow-up EXEC
Help     ==>                                     Associated HELP Panel
Targets  ==>                                     Target System
Udata    ==>                                     User Data
Origin   ==>                                     Origin
User     ==>                                     Userid
Alarm    ==>                                     Sound Alarm (YES/NO)
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 14 The Rule in this example does not issue an ALERT; press **Enter**.

The Alert Action(s) II - MQS panel is displayed.

```

BMC Software ----- Alert Action(s) II - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE
          Rule-set === RULMQS01                 Rule-id === APQHI
Retain   ==>                                     Yes/No
Escalate Direction ==>                             Up/Down
          Interval ==>
          ==>
          ==>
          ==>
          ==>
          ==>
          Disposition ==>                             Keep/Delete
          EXEC      ==>
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 15 The Rule in this example does not issue an ALERT; press **Enter**.

The Rule Processor Detail Control panel is redisplayed.

Figure 5-13 Rule Processor Detail Control Panel (Example 3)

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>                                                    TGT --- SYSE
                                                                TIME --- 18:24:24
                                                                DATE --- 00/11/15

The following options are displayed in sequence, or may
be selected by entering the two-character code

S1 - Selection Criteria          A1 - Action Specification
SV - Variable Dependencies      AA - Alert Actions(s)

Rule ID      ==> APQHI
Event Type   ==> MQS      Type of event ( ? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>          (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>          (Interval length, 1-99999 seconds)
then status  ==>          (SUSPEND, DISABLE)

Application information:
Group        ==> MQ      Function      ==> MONTRMQ Code ==> R2
Author       ==> BMCUSER Description    ==> MONITOR AP QUEUES
Last Modified by      on      at
    
```

Step 16 Press PF3 to apply changes.

The Rule Set Overview panel is redisplayed.

Figure 5-14 Rule Set Overview Panel (Example 2)

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==>                                                    TGT --- SYSE
Rule Set ID: RULMQS01      Ruleset Strategy ==> ALL          DATE --- 00/11/16
Primary commands: Add, Save, Sort, Unsort, Reset, Filter      TIME --- 18:32:54
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:                                                    right/left

LC  Rule-id Stat Text-id          Type   Fired EXEC          Changed          ID
---  ---  ---  ---  ---  ---  ---  ---  ---  ---
___  MQQDPLOW ENA Q_DEPTH_LOW      MQS    0  MQVARS 1  00/11/15 15:56 BAOJDB4
___  MQQDPLO2 ENA Q_DEPTH_LOW      MQS    0  MQVARS 2  00/11/15 15:56 BAOJDB4
___  MQSVINTH ENA Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15 15:56 BAOJDB4
___  MQSVINT2 ENA Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15 15:56 BAOJDB4
___  MQSVINOK ENA Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15 15:56 BAOJDB4
___  MQSVINO2 ENA Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15 15:56 BAOJDB4
___  MQPUTENA ENA PUT_INHIBITED  MQS    0  MQVER MQ  97/02/19 14:51 JDB3
___  MQPUTEN2 SUS PUT_INHIBITED  MQS    0  MQVER MQ  97/02/19 14:51 JDB3
___  MQQFULL  ENA Q_FULL          MQS    0  MQVER MQ  97/02/19 20:49 JDB3
___  QUEUE1   ENA                   MQS    0  VARDISP 00/11/15 15:56 BAOJDB4
___  APQHI    ENA Q_DEPTH_HIGH    MQS    0                   00/11/15 18:32 BAOJDB4
***** END OF DATA *****
    
```

Note: The Text-id for the new Rule APQHI is Q_DEPTH_HIGH.

The Rule is enabled and will begin to fire when the Q_DEPTH_HIGH event occurs for the queue PROD.AP within the CSQ1 queue manager. However, at this point the Rule is not yet saved to disk.

Step 17 To save the Rule, type the **SAVE** primary command on the **COMMAND** line.

Warning! If you do not type the **SAVE** primary command, the following warning is displayed when you press **PF3**:

Figure 5-15 Confirming Rule Set Modifications Panel (Example 1)

```

BMC Software ----- Confirm Rule Set Modifications ----- AutoOPERATOR
COMMAND ==>                                                    TGT --- SYSE
+-----+
+ WARNING! Changes made to Rule Set RULMQS01 have not been saved. Those +
+ changes were one or more of the following:                          +
+                                                                    +
+ o A Rule was changed.                                             +
+ o The status of a Rule was modified.                             +
+ o A Rule was added, deleted, inserted, or copied.                +
+ o A Rule was moved.                                              +
+ o The individual Rule Set strategy changed.                       +
+                                                                    +
+ Please do one of the following:                                    +
+                                                                    +
+ - Enter SAVE to save RULMQS01 to the BBIPARM dataset.            +
+ - Enter NOSAVE to exit WITHOUT saving RULMQS01 to the BBIPARM dataset. +
+ - Press END to return to Rule Set Overview.                      +
+-----+
    
```

The options are as follows:

- Type **SAVE** to save the Rule.
- Type **NOSAVE** to cancel saving the newly created Rule to disk, leaving the memory copy intact.
- Press **PF3** to return to the Rule Set Overview panel.

Creating a Rule for a Non-MQFMT_EVENT Message

This section shows you how to create a MAINVIEW AutoOPERATOR Rule that will listen for and fire when a MQSeries message (with the format MQFMT_STRING and the message CLOSE ALL included in the first 255 bytes of the message text) is written to the PROD.AP.QUEUE queue.

Note: Be aware that the variable IMFTEXT that is passed to an EXEC when scheduled from a Rule, has the same limitations (255 bytes) as the MSG Buffer of the Selection Criteria Panel. For more information regarding IMFTEXT, see Chapter 4 in the *MAINVIEW AutoOPERATOR Advanced Automation Guide*. To access data in the message beyond 255 bytes, an EXEC must be used. By using an EXEC, you can access up to 32,767 bytes of the message content.

During initialization or during processing of the .RESET MQ command, MAINVIEW AutoOPERATOR reads through monitored queues and establishes a browse cursor after the last message is read. If a message of higher priority than the last message read is put onto the queue, MAINVIEW AutoOPERATOR Rule processing is notified of the message arrival but will not be able to get the message from the queue, and therefore will not be able to automate it. You will know that this situation might have occurred when you see a QM6542E message issued to the BBI journal. To prevent this situation from happening, you can verify that all messages going to the monitored queue are of the same priority. If verification is not possible and the problem occurs, you can access the message by invoking an EXEC that can do a GET for the message.

By following the example, you will learn how to create a Rule to listen for this message. When the Rule fires, it will update a variable that indicates what happened.

To create the Rule, you must enter the correct Selection Criteria information. The following table describes the information you need to enter.

You Need to Know	Which Is	To Enter It on This Panel
The Rule event type	MQS	Rule Processor Detail Control
The queue manager name	CSQ1, in this example	Selection Criteria - MQS
The message format	MQFMT_STRING	Selection Criteria - MQS
The name of the queue to which the message was PUT	PROD.AP.QUEUE	Selection Criteria - MQS
The message content	close all	Selection Criteria - MQS

Example: Creating a Non-MQFMT_EVENT Message Rule MQSeries Message

To create a Rule that fires when a MQSeries message is written to the PROD.AP.QUEUE queue with a message text of CLOSE ALL, follow these steps:

- Step 1** From the Automation Control panel, type the E line command, for (E)nable, and press **Enter** to enable the Rule Set you want to add the Rule to. You can add Rules only to enabled Rule Sets; see Figure 5-16.

Figure 5-16 Creating a Rule for a Message with Format MQFMT_STRING: Automation Control Panel (Example 1)

```

BMC Software ----- Automation Control ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> SYSE
Primary commands: Add                             DATE --- 00/11/16
                                                    TIME --- 14:14:39

Automation Status      ==> ACTIVE                (Active, Inactive)
Automation Strategy    ==> INDIVIDUAL            (Individual, All, First, Qualified)
Honor MPF Suppression  ==> NO                   (NO/YES)

Automation Statistics
Total Events           48,538  Display suppressed           67
Events Handled         22,407  Hardcopy suppressed           0
Current arrival rate   2 / sec  Rule generated Alerts      22,223
Peak arrival rate     99 / sec  Rule invoked EXECs         276

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (SA)ve
           (M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC  Rule-Set Status  Rules  Fired  Filtered  Date      Time      Strategy
___ AAORUL00 ENABLED   15    22,223  48,573  00/11/16  08:18:59  FIRST
___ AAORULBC ENABLED   54     188    48,573  00/11/16  08:18:59  FIRST
___ AAORULCM ENABLED   52     59     48,573  00/11/16  08:18:59  FIRST
___ RULCICS ENABLED    25     1      48,573  00/11/16  08:19:00  FIRST
e_  RULMQS01 DISABLED   8      0       38     00/11/16  14:14:38  FIRST
***** END OF DATA *****
    
```

- Step 2** To select the Rule Set, type the S line command, for (S)elect, in the LC field next to the Rule Set name and press **Enter**. See Figure 5-17 on page 5-23

Figure 5-17 Creating a Rule for a Message with Format MQFMT_STRING: Automation Control Panel (Example 2)

```

BMC Software ----- Automation Control ----- RULMQS01 ENABLED
COMMAND ==>                                     TGT ==> SYSE
Primary commands:                                DATE --- 00/11/16
                                                TIME --- 08:35:54

Automation Status      ==> ACTIVE                (Active, Inactive)
Automation Strategy    ==> INDIVIDUAL            (Individual, All, First, Qualified)
Honor MPF Suppression ==> NO                    (NO/YES)

Automation Statistics
Total Events           48,538  Display suppressed           67
Events Handled         22,407  Hardcopy suppressed           0
Current arrival rate   2 / sec  Rule generated Alerts    22,223
Peak arrival rate      99 / sec  Rule invoked EXECs       276

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (SA)ve
           (M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC  Rule-Set Status  Rules   Fired  Filtered  Date      Time      Strategy
___ AAORUL00 ENABLED   15     22,223  48,573  00/11/16  08:18:59  FIRST
___ AAORULBC ENABLED  54      188    48,573  00/11/16  08:18:59  FIRST
___ AAORULCM ENABLED  52       59    48,573  00/11/16  08:18:59  FIRST
___ RULCICS ENABLED   25       1     48,573  00/11/16  08:19:00  FIRST
s__ RULMQS01 ENABLED   8        0      38     00/11/16  14:14:38  FIRST
***** END OF DATA *****
    
```

The Rule Set Overview panel is displayed. Figure 5-18 shows an example of a Rule Set named RULMQS01 and the Rules it contains.

Step 3 To add a new Rule, use the **ADD** primary command on the **COMMAND** line.

Figure 5-18 Creating a Rule for a Message with Format MQFMT_STRING: Rule Set Overview Panel (Example 1)

```

BMC Software ----- Rule Set Overview ----- AutoOPERATOR
COMMAND ==> add                                     TGT --- SYSE
Rule Set ID: RULMQS01  Ruleset Strategy ==> ALL      DATE --- 00/11/16
Primary commands: Add, Save, Sort, Unsort, Reset, Filter  TIME --- 09:34:54
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
           (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)epet
Sort Criterion:                                       right/left

LC  Rule-id Stat Text-id      Type   Fired EXEC      Changed      ID
___ MQQDPLOW ENA Q_DEPTH_LOW   MQS    0  MQVARS 1  00/11/15  15:56  BAOJDB4
___ MQQDPLO2 ENA Q_DEPTH_LOW   MQS    0  MQVARS 2  00/11/15  15:56  BAOJDB4
___ MQSVINTH ENA Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15  15:56  BAOJDB4
___ MQSVINT2 ENA Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15  15:56  BAOJDB4
___ MQSVINOK ENA Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15  15:56  BAOJDB4
___ MQSVINO2 ENA Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15  15:56  BAOJDB4
___ MQPUTENA ENA PUT_INHIBITED  MQS    0  MQVER MQ  00/11/15  14:51  JDB3
___ MQPUTEN2 SUS PUT_INHIBITED  MQS    0  MQVER MQ  00/11/15  14:51  JDB3
___ MQQFULL  ENA Q_FULL        MQS    0  MQVER MQ  00/11/15  20:49  JDB3
___ QUEUE1   ENA                MQS    0  VARDISP 00/11/15  15:56  BAOJDB4
___ APQHI    ENA Q_DEPTH_HIGH   MQS    0                00/11/15  18:32  BAOJDB4
***** END OF DATA *****
    
```

The Rule Processor Detail Control panel is displayed, as shown in Figure 5-19.

Figure 5-19 **Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 1)**

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>>>                                     TGT --- SYSE
                                                    TIME --- 09:35:54
                                                    DATE --- 00/11/16

The following options are displayed in sequence, or may
be selected by entering the two-character code
S1 - Selection Criteria           A1 - Action Specification
SV - Variable Dependencies        AA - Alert Actions(s) I
                                   AD - Alert Actions(s) II

Rule ID           ==>>>
Event Type        ==>>>           Type of event ( ? for list)
Initial Mode      ==>>>           (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched        ==>>>           (Maximum # times matched within INTERVAL, 1-100)
in seconds        ==>>>           (Interval length, 1-99999 seconds)
then status       ==>>>           (SUSPEND, DISABLE)

Application information:
Group             ==>>>           Function           ==>>>           Code           ==>>>
Author            ==>>>           Description        ==>>>
Last Modified by  on               at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes
    
```

Step 4 Type the following information:

- a unique Rule ID in the Rule ID field (for example, CLOSAP); see Figure 5-20
- **MQS** in the **Event Type** field; see Figure 5-20 on page 5-25.

Note: The Event Type MQS will be accepted *only* if the MAINVIEW AutoOPERATOR for MQSeries key is present. If the key is not present, you receive a short error message in the upper right corner of the panel.

Figure 5-20 Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 2)

```

BMC Software -----Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE
                                                TIME --- 15:46:37
                                                DATE --- 00/11/16

The following options are displayed in sequence, or may
be selected by entering the two-character code
  S1 - Selection Criteria           A1 - Action Specification
  SV - Variable Dependencies       AA - Alert Actions(s) I
                                   AD - Alert Actions(s) II

Rule ID      ==> CLOSAP
Event Type   ==> MQS      Type of event (? for list)
Initial Mode ==> ENABLED (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched   ==>          (Maximum # times matched within INTERVAL, 1-100)
in seconds   ==>          (Interval length, 1-99999 seconds)
then status  ==>          (SUSPEND, DISABLE)

Application information:
Group        ==>          Function      ==>          Code   ==>
Author       ==>          Description   ==>
Last Modified by on      at

Press ENTER to continue, END to apply changes, CANCEL to cancel changes
    
```

Step 5 Press Enter.

The Selection Criteria - MQS panel is displayed:

Figure 5-21 Creating a Rule for a Message with Format MQFMT_STRING: Selection Criteria - MQS Panel (Example 1)

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE

Rule-set == RULMQS01           Rule-id == CLOSAP

Queue Identification: (1 to 12 Queue Managers)
Manager(s) ==>
Queue Id    ==>

Message Identification:
Format      ==>          (Value from MD Format field)
Event Type  ==>          (Enter ? for help)

Sub  Len  Op  Value
Msgid ==> ___ : ___ ___ _____
CorrelId ==> ___ : ___ ___ _____
Msg Buffer ==> ___ : ___ ___ _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 6 To enter the selection criteria for this event

- 6.A** In the **Manager(s)** field, enter the name of your queue manager (for example, CSQ1); see Figure 5-22.
 - 6.B** In the **Queue ID** field, type **PROD.AP.QUEUE**. This is the name of the queue where this message must originate; see Figure 5-22.
 - 6.C** In the **Format** field, type **MQFMT_STRING**; see Figure 5-22.
- This is the message format. When you create Rules for MQSeries messages with a format other than MQFMT_EVENT, use the fields **MsgId**, **CorrelId** and **Msg Buffer** to add additional selection criteria, if required.
- 6.D** In the **Msg Buffer** field, type **CLOSE ALL**.

The message must match this text; see Figure 5-22.

Note: Be aware that the variable IMFTEXT, which is passed to an EXEC when scheduled from a Rule, has the same limitations (255 bytes) as the **MSG Buffer** of the Selection Criteria Panel. For more information regarding IMFTEXT, see Chapter 4 in the *MAINVIEW AutoOPERATOR Advanced Automation Guide*. To access data in the message beyond 255 bytes, you must use an EXEC. Using an EXEC, you can access up to 32,767 bytes of the message content.

Figure 5-22 Selection Criteria - MQS Panel (Example 2)

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==> TGT --- SYSE

      Rule-set === RULMQS01          Rule-id === CLOSAP

Queue Identification: (1 to 12 Queue Managers)
Manager(s)  ==> CSQ1
Queue Id    ==> PROD.AP.QUEUE

Message Identification:
Format      ==> MQFMT_STRING          (Value from MD Format field)
Event Type  ==>                      (Enter ? for help)

      Sub  Len  Op  Value
Msgid    ==> ___ : ___ ___ _____
CorrelId   ==> ___ : ___ ___ _____
Msg Buffer  ==> ___ : ___ ___ CLOSE ALL _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```


Step 9 Use the Action Specification - MQS panel to specify the action for the Rule to take when it fires. For this example, the two actions this Rule takes are start a Started Task named APCLOSE and delete the message from the queue.

To specify the actions for this Rule:

9.A Change **Cmd(Type MQ)** to **Cmd (Type OS/390)** and type **#S APCLOSE** in the **Cmd** field; see Figure 5-24.

9.B Type **NO** in the Keep Message field, which deletes the original message; see Figure 5-24.

Figure 5-24 Action Specification - MQS Panel (Example 2)

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==> TGT --- SYSE

Rule-set === RULMQS01 Rule-id === CLOSAP

Automation Actions:
Journal ==>
EXEC Name/Parms ==>
Cmd (Type MVS ) ==> #S APCLOSE

Set Variable ==> ==>
Notify ==> Outboard Pager ID
Info ==>

DOM Id ==> Delete Operator Message
Issue WTO Msg ==>

MQSeries Automation Actions:
Keep Message ==> NO (Yes or NO) Remove DLH ==> (Yes or No)
Destination Que ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 10 Press **Enter**.

The Alert Action(s) I -MQS panel is displayed, as shown in the next figure.

```

BMC Software ----- Alert Action(s) I - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE
          Rule-set === RULMQS01                 Rule-id  === CLOSAP
Function ==>                                     (ADD, DELETE, DELETEQ)
Key      ==>
Text     ==>
Queues   ==>                                     Alert Queue Name(s)
Priority ==>                                     (CRIT,MAJ,MIN,WARN,INFO,CLEAR)
Color    ==>                                     RED,PINK,YEL,DKBL,LTBL,GRE,WHI
PCMD     ==>
System   ==>                                     Return to target after PCMD
EXEC     ==>                                     Follow-up EXEC
Help     ==>                                     Associated HELP Panel
Targets  ==>                                     Target System
Udata    ==>                                     User Data
Origin   ==>                                     Origin
User     ==>                                     Userid
Alarm    ==>                                     Sound Alarm (YES/NO)
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 11 This example does not issue an ALERT; press **Enter**.

The Alert Action(s) II - MQS panel is displayed:

```

BMC Software ----- Alert Action(s) II - MQS ----- AutoOPERATOR
COMMAND ==>                                     TGT --- SYSE
          Rule-set === RULMQS01                 Rule-id  === CLOSAP
Retain   ==>                                     Yes/No
Escalate ==>                                     Up/Down
  Direction ==>
    Interval ==>
              ==>
              ==>
              ==>
              ==>
              ==>
  Disposition ==>                                     Keep/Delete
    EXEC      ==>
Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Step 12 This example does not issue an ALERT; press **Enter**.

The Rule Processor Detail Control Panel is redisplayed, as shown in Figure 5-25 on page 5-30.

Figure 5-25 Creating a Rule for a Message with Format MQFMT_STRING: Rule Processor Detail Control Panel (Example 3)

```

BMC Software ----- Rule Processor Detail Control ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE
                                                    TIME --- 15:23:36
                                                    DATE --- 00/11/16

The following options are displayed in sequence, or may
be selected by entering the two-character code

S1 - Selection Criteria           A1 - Action Specification
SV - Variable Dependencies        AA - Alert Action(s) I
                                   AD - Alert Action(s) II

Rule ID       ==> CLOSAP
Event Type    ==> MQS      Type of event ( ? for list)
Initial Mode  ==> ENABLED  (ENABLED/DISABLED/TEST)

Criteria match rate:
If matched    ==>          (Maximum # times matched within INTERVAL, 1-100)
in seconds    ==>          (Interval length, 1-99999 seconds)
then status   ==>          (SUSPEND, DISABLE)

Application information:
Group         ==>          Function      ==>          Code      ==>
Author        ==> BAOJDB4  Description  ==>
Last Modified by      on          at
    
```

Step 13 Press PF3 to apply the changes.

The Rule Set Overview Panel is redisplayed:

Figure 5-26 Rule Set Overview Panel (Example 2)

```

BMC Software ----- Rule Set Overview ----- RULE UPDATED
COMMAND ==>                                           TGT --- SYSE
Rule Set ID: RULMQS01   Ruleset Strategy ==> ALL      DATE --- 00/11/16
Primary commands: Add, Save, Sort, Unsort, Reset, Filter  TIME --- 16:06:12
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
              (C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion:                                           right/left

LC  Rule-id  Stat  Text-id          Type  Fired EXEC      Changed      ID
---  ---    ---    ---              ---    ---  ---    ---        ---
___  MQQDPLOW  ENA   Q_DEPTH_LOW      MQS    0  MQVARS 1  00/11/15 15:56  BAOJDB4
___  MQQDPLO2  ENA   Q_DEPTH_LOW      MQS    0  MQVARS 2  00/11/15 15:56  BAOJDB4
___  MQSVINTH  ENA   Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15 15:56  BAOJDB4
___  MQSVINT2  ENA   Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15 15:56  BAOJDB4
___  MQSVINOK  ENA   Q_SERVICE_INTERV MQS    0  MQVARS 1  00/11/15 15:56  BAOJDB4
___  MQSVINO2  ENA   Q_SERVICE_INTERV MQS    0  MQVARS 2  00/11/15 15:56  BAOJDB4
___  MQPUTENA  ENA   PUT_INHIBITED    MQS    0  MQVER  MQ  97/02/19 14:51  JDB3
___  MQPUTEN2  SUS   PUT_INHIBITED    MQS    0  MQVER  MQ  97/02/19 14:51  JDB3
___  MQQFULL   ENA   Q_FULL           MQS    0  MQVER  MQ  97/02/19 20:49  JDB3
___  QUEUE1    ENA                               MQS    0  VARDISP 00/11/15 15:56  BAOJDB4
___  APQHI     ENA   Q_DEPTH_HIGH     MQS    0          00/11/15 18:32  BAOJDB4
___  CLOSAP    ENA                               MQS    0          00/11/16 16:06  BAOJDB4
***** END OF DATA *****
    
```

The Rule is enabled and fires when a MQFMT_STRING message with the CLOSE ALL message text is detected for the PROD.AP.QUEUE queue within the CSQ1 queue manager. However, at this point, the Rule is not saved to disk.

Note: Be aware that the variable IMFTEXT, which is passed to an EXEC when scheduled from a Rule, has the same limitations as the MSG Buffer of the Selection Criteria Panel, 255 bytes. For more information regarding IMFTEXT, see Chapter 4 in the *MAINVIEW AutoOPERATOR Advanced Automation Guide*. To access data in the message beyond 255 bytes, you must use an EXEC. Using an EXEC, you can access up to 32,767 bytes of the message content.

Step 14 To save the Rule, type the **SAVE** primary command on the command line.

If you do not type the **SAVE** primary command, the warning shown in Figure 5-27 is displayed when you press **PF3**:

Figure 5-27 Confirming Rule Set Modifications Panel (Example 2)

```

BMC Software ----- Confirm Rule Set Modifications ----- AutoOPERATOR
COMMAND ==>                                                    TGT --- SYSE
+-----+
+ WARNING! Changes made to Rule Set RULMQS01 have not been saved. Those +
+ changes were one or more of the following:                          +
+                                                                     +
+ o A Rule was changed.                                             +
+ o The status of a Rule was modified.                              +
+ o A Rule was added, deleted, inserted, or copied.                 +
+ o A Rule was moved.                                               +
+ o The individual Rule Set strategy changed.                        +
+                                                                     +
+ Please do one of the following:                                     +
+                                                                     +
+ - Enter SAVE to save RULMQS01 to the BBIPARM dataset.             +
+ - Enter NOSAVE to exit WITHOUT saving RULMQS01 to the BBIPARM dataset. +
+ - Press END to return to Rule Set Overview.                        +
+-----+
    
```

The options are as follows:

- Type **SAVE** to save the Rule.
- Type **NOSAVE** to cancel saving the newly created Rule to disk, but leaving memory copy intact.
- Press **End** to return to the Rule Set Overview panel.

MQSeries Instrumentation Events

This section describes the MQSeries instrumentation events (MQFMT_EVENT) for which you can create Rules.

In MQSeries, an instrumentation event is a logical combination of conditions that is detected by a queue manager or channel instance. The result of such an event is that the queue manager or channel instance puts a special message, called an event message, on an event queue.

For more information about MQFMT_EVENT messages, refer to “What MQSeries Events Are” on page 1-5.

Table 5-2 shows a list of valid instrumentation events that you can automate through the use of Rules. Enter a ? in the **Event Type** field to get a list of the supported event types.

- The third column lists the MQSeries-specific MAINVIEW AutoOPERATOR variables that are associated with each instrumentation event.
- The fourth column shows the instrumentation event category.
- The fifth column shows the queue manager attribute that enables or disables generation of the instrumentation event.

Table 5-2 MQSeries Instrumentation Events (Part 1 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
alias_base_q_type_error	a MQOPEN or MQPUT1 call was issued specifying an alias queue as the destination, but the BaseQName in the alias queue definition resolves to a queue that is not a local queue, or local definition of a remote queue	<ul style="list-style-type: none"> • IMFQ_EVENT_BASEQNAME: Queue manager name (to which the alias resolves) • IMFQ_EVENT_QNAME: Name of queue from object descriptor • IMFQ_EVENT_QTYPE: Type of queue to which the alias resolves (A for alias, M for model) • IMFQ_EVENT_APPLTYPE: Type of application that caused the event. For example: 	Queue Manager	localev

Table 5-2 MQSeries Instrumentation Events (Part 2 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
alias_base_q_type_error continued	a MQOPEN or MQPUT1 call was issued specifying an alias queue as the destination, but the BaseQName in the alias queue definition resolves to a queue that is not a local queue, or local definition of a remote queue	MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE <ul style="list-style-type: none"> • MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote 	Queue Manager	localev
bridge_started	the IMS bridge has been started	IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote IMFQ_EVENT_BRIDGETYPE: Only current value is 1 (OTMA) IMFQ_EVENT_BRIDGENAME: Name of bridge	Channel	n/a

Table 5-2 MQSeries Instrumentation Events (Part 3 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
bridge_stopped	the IMS bridge has been stopped	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_BRIDGETYPE: Only current value is 1 (OTMA) • IMFQ_EVENT_BRIDGENAME: Name of bridge • IMFQ_EVENT_REASONQUALIFIER : Reason code: 11 - Bridge stopped, one side or the other issued a normal IXCLEAVE request. -12 - Bridge stopped, but an error was reported. • IMFQ_EVENT_ERRORIDENTIFIER: Error identification 	Channel	n/a
channel_activated	<p>a channel which was waiting to become active has now become active</p> <p>Not generated if the channel did not have to wait for a slot to be released.</p> <p>Not produced for CICS for distributed queue management in MQSeries for OS/390.</p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_CHANNELNAME: Channel name • IMFQ_EVENT_XMITQNAME: Transmission queue name • IMFQ_EVENT_CONNECTIONNAME Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition. 	Channel	n/a

Table 5-2 MQSeries Instrumentation Events (Part 4 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
channel_conv_error	<p>a channel is unable to perform data conversion and the MQGET call to get a message from the transmission queue resulted in a data conversion error</p> <p>The reason for the failure is identified by IMFQ_EVENT_ConversionReasonCode.</p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_CHANNELNAME: Channel name • IMFQ_EVENT_XMITQNAME: Transmission queue name • IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition. • IMFQ_EVENT_FORMAT: Name of format • IMFQ_EVENT_REASONQUALIFIER <ul style="list-style-type: none"> --Conversion reason code: MQRC_FORMAT_ERROR MQRC_SOURCE_CCSID_ERROR MQRC_SOURCE_INTEGER_ENC_ERROR MQRC_SOURCE_DECIMAL_ENC_ERROR MQRC_SOURCE_FLOAT_ENC_ERROR MQRC_TARGET_CCSID_ERROR MQRC_TARGET_INTEGER_ENC_ERROR MQRC_TARGET_DECIMAL_ENC_ERROR MQRC_TARGET_FLOAT_ENC_ERROR MQRC_NOT_CONVERTED MQRC_CONVERTED_MSG_TOO_BIG MQRC_TRUNCATED_MSG_ACCEPTED MQRC_TRUNCATED_MSG_FAILED 	Channel	n/a

Table 5-2 MQSeries Instrumentation Events (Part 5 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
channel_not_activated	<p>a channel is waiting to become active because the limit on active channels has been reached</p> <p>The channel waits until an active slot is released when another channel ceases to be active. A CHANNEL_ACTIVATED event will then be generated.</p> <p><i>Not produced for CICS for distributed queue management in MQSeries for OS/390.</i></p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_CHANNELNAME: Channel name • IMFQ_EVENT_XMITQNAME: Transmission queue name • IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition. 	Channel	n/a
channel_started	<p>a channel was started because</p> <ul style="list-style-type: none"> • A Start Channel operator command was issued • An instance of a channel connection has been established <p><i>Not produced for CICS for distributed queue management in MQSeries for OS/390.</i></p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_CHANNELNAME: Channel name • IMFQ_EVENT_XMITQNAME: Transmission queue name • IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition. 	Channel	n/a
channel_stopped	channel stopped	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_CHANNELNAME: Channel name • IMFQ_EVENT_XMITQNAME: Transmission queue name • IMFQ_EVENT_CONNECTIONNAME: Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition. 	Channel	n/a

Table 5-2 MQSeries Instrumentation Events (Part 6 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
channel_ stopped	channel stopped	<ul style="list-style-type: none"> • IMFQ_EVENT_REASONQUALIFIER -- Reason qualifier: MQRQ_CHANNEL_STOPPED_OK MQRQ_CHANNEL_STOPPED_ERROR MQRQ_CHANNEL_STOPPED_RETRY MQRQ_CHANNEL_STOPPED_DISABLED • IMFQ_EVENT_ERRORIDENTIFIER: See the section regarding the CHANNEL_STOPPED event, field Error identifier, in the IBM publication <i>MQSeries Programmable System Management</i> for a description of this data. • IMFQ_EVENT_AUXERRORDATAIN T1: First integer of auxiliary error data for channel errors • IMFQ_EVENT_AUXERRORDATAIN T2: Second integer of auxiliary error data for channel errors • IMFQ_EVENT_AUXERRORDATAS TR1: First string of auxiliary error data for channel errors • IMFQ_EVENT_AUXERRORDATAS TR2: Second string of auxiliary error data for channel errors • IMFQ_EVENT_AUXERRORDATAS TR3: Third string of auxiliary error data for channel errors 	Channel	n/a

Table 5-2 MQSeries Instrumentation Events (Part 7 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
def_xmit_q_type_error	<p>a MQOPEN or MQPUT1 was issued specifying a remote queue as the destination</p> <p>Either a local definition of the remote queue was specified, or a queue manager alias was being resolved, but in either case the XmitQName attribute in the local definition is blank. No transmission queue is defined with the same name as the destination queue manager, so the local queue manager has attempted to use the default transmission queue. However, there is a queue defined by the DefXmitQName queue manager attribute, but it is not a local queue.</p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for that caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_XMITQNAME: Name of the transmission queue • IMFQ_EVENT_QTYPE: Type of queue (A for alias, R for remote) 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 8 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
def_xmit_q_usage_error	A MQOPEN or MQPUT1 was issued specifying a remote queue as the destination. Either a local definition of the remote queue was specified, or a queue manager alias was being resolved, but in either case the XmitQName attribute in the local definition is blank. No transmission queue is defined with the same name as the destination queue manager, so the local queue manager has attempted to use the default transmission queue. However, the queue defined by the XmitQName queue manager attribute does not have a usage attribute of MQUS_TRANSMISSION.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_XMITQNAME: Name of the transmission queue 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 9 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
get_inhibited	An attempt was made to retrieve a message (using MQGET) but the retrieval was inhibited for the queue.	<ul style="list-style-type: none"> • IMFQ_EVENT_QNAME: Name of queue which GETs are inhibited • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_APPLTYPE: Type of application that caused event. <p style="margin-left: 20px;">For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT</p>	Queue Manager	inhibitv

Table 5-2 MQSeries Instrumentation Events (Part 10 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
not_authorized type 1,2,3,4	<p>One of four codes is possible:</p> <ol style="list-style-type: none"> 1. On a MQCONN call, the user is not authorized to connect to the queue manager 2. On a MQOPEN/MQPUT 1: the user is not authorized to open the object for the option(s) specified 3. On a MQCLOSE: the user is not authorized to delete the object, which is a permanent dynamic queue, and the HOBJ parameter specified on the MQCLOSE call is not the handle returned by the MQOPEN call which created the queue 4. A command has been issued from a user ID that is not authorized to access the object specified in the command 	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_REASONQUALIFIER : Reason code (type of auth error, 1,2,3 or 4) • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OPTIONS: Open options used during the MQOPEN call (type 2 only) • IMFQ_EVENT_QNAME: Name of queue • IMFQ_EVENT_COMMAND: Identifier for the command (type 4 only). See the discussion for PCF header in the IBM publication <i>MQSeries Programmable System Management</i>. • IMFQ_EVENT_USERIDENTIFIER: User identification that caused the authorization check. 	Queue Manager	authorev

Table 5-2 MQSeries Instrumentation Events (Part 11 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
not_authorized type 1,2,3,4 continued		<ul style="list-style-type: none"> • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager (type 2 only) • IMFQ_EVENT_PROCESSNAME: Name of the process whose attributes have changed (type 2 only) 		
put_inhibited	an attempt was made to send a message to a queue (using MQPUT) but the attempt was inhibited for the queue	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_QNAME: Name of queue for which PUTs are inhibited • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager if the object queue manager (when the queue was opened) is not the queue manager currently connected 	Queue Manager	inhibtev

Table 5-2 MQSeries Instrumentation Events (Part 12 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
q_depth_high	a MQPUT or MQPUT1 call caused the queue depth to increase to the queue depth high limit	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the statistics were last reset • IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset • IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset • IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset 	Performance	perfmev
q_depth_low	A MQGET call caused the queue depth to decrease to the queue depth low limit.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the statistics were last reset • IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset • IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset • IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset 	Performance	perfmev

Table 5-2 MQSeries Instrumentation Events (Part 13 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
q_full	A MQPUT or MQPUT1 call failed because the queue is full.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the statistics were last reset • IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset • IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset • IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset 	Performance	perfmev
q_mgr_active	Detected when a queue manager becomes active. In MQSeries for OS/390, this event is not generated for the first start of a queue manager, only on subsequent restarts.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote 	Queue Manager	strstpev
q_mgr_not_active	Detected when a queue manager is requested to stop or quiesce.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_REASONQUALIFIER: Reason code (5 for queue manager stopping, 6 for queue manager quiescing) 	Queue Manager	strstpev

Table 5-2 MQSeries Instrumentation Events (Part 14 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
q_service_interval_high	No successful gets or puts have been detected within an interval that is greater than the threshold specified in the queue service interval attribute.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_TIMESINCERESET: Time (in seconds) since the statistics were last reset; for this event, this is greater than the service interval • IMFQ_EVENT_HIGHQDEPTH: Max number of message on the queue since the statistics were last reset • IMFQ_EVENT_MSGENQCOUNT: Number of messages enqueued since statistics were last reset • IMFQ_EVENT_MSGDEQCOUNT: Number of messages dequeued since statistics were last reset 	Performance	perfmev

Table 5-2 MQSeries Instrumentation Events (Part 15 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
q_type_error	<p>Could result for one of the following reasons:</p> <ul style="list-style-type: none"> On a MQOPEN call, the 'object queue manager name' field specifies the name of a local definition of a remote queue, and in the definition the 'remote queue manager name' the attribute is the name of the local queue manager A MQPUT1 call specifies the name of a model queue On a previous MQPUT/MQPUT1 call, the 'reply to queue' specified a model queue 	<ul style="list-style-type: none"> IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote IMFQ_EVENT_QNAME: Name of queue for which caused the error IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT IMFQ_EVENT_APPLNAME: Name of application IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 16 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
remote_q_name_error	<p>During a MQOPEN or MQPUT1 call, one of the following occurred:</p> <ul style="list-style-type: none"> • A local definition of a remote queue (or an alias to one) was specified, but the 'remote queue name' attribute in the remote queue definition is blank • A MQPUT1 call specifies the name of a model queue • The 'object queue manager name' field was not blank, and not the name of the local queue manager, but the 'object name' is blank 	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 17 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
unknown_ alias_base_q	a MQOPEN or MQPUT1 was issued specifying an alias queue as the target, but the 'base queue name' in the alias queue attributes are not recognized.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_BASEQNAME: Queue name to which the alias resolves 	Queue Manager	localev

Table 5-2 MQSeries Instrumentation Events (Part 18 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
UNKNOWN_DEF_XMIT_Q	<p>a MQOPEN or MQPUT1 was issued specifying a remote queue. If a local definition of the remote queue was specified, or if a queue manager alias is being resolved, the XmitQName attribute in the local definition is blank. No queue is defined with the same name as the destination queue manager. The queue manager has therefore attempted to use the default transmission queue. However, the name defined by the DefXmitQName queue manager attribute is not the name of a locally defined queue.</p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. <p>For example:</p> <p>MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT</p> <ul style="list-style-type: none"> • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_XMITQNAME: Name of the transmission queue 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 19 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
unknown_ object_name	<p>a MQOPEN or MQPUT1 was issued and the 'object queue manager name' is set to</p> <ul style="list-style-type: none"> • Blank • Name of the local queue manager <p>The name of a local definition of a remote queue in which the 'remote queue manager name' attribute is the name of the local queue manager.</p>	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. <p>For example:</p> <p>MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT</p> <ul style="list-style-type: none"> • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_PROCESSNAME: Name of the process (application) issuing the MQI call that caused the event 	Queue Manager	localev

Table 5-2 MQSeries Instrumentation Events (Part 20 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
unknown_ remote_q_mgr	A MQOPEN or MQPUT1 was issued, and an error occurred with the queue name resolution. For more information, refer to the IBM publication <i>MQSeries Programmable System Management</i> .	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 21 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
unknown_xmit_q	On a MQOPEN or MQPUT1, a message is to be sent to a remote queue manager. The ObjectName or the ObjectQMGrName in the object descriptor specifies the name of a local definition of a remote queue (in the latter case queue manager aliasing is being used), but the XmitQName attribute of the definition is not blank and not the name of a locally defined queue.	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_XMITQNAME: Name of the transmission queue 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 22 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
xmit_q_type_error	<p>Occurs during a MQOPEN or MQPUT1 call when a message is directed to be sent to a remote queue manager. The 'object name' or 'object queue manager name' field specifies the local definition of a remote queue, but one of the following applies to the 'xmit name' attribute of the definition:</p> <ul style="list-style-type: none"> • 'Xmit queue name' specifies a queue that is not a local queue • 'Xmit queue name' is blank, but 'remote queue manager name' specifies a queue that is not a local queue 	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_XMITQNAME: Transmission queue name, not available for some events • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_QTYPE: Type of queue (A for alias, R for remote) 	Queue Manager	remoteev

Table 5-2 MQSeries Instrumentation Events (Part 23 of 23)

Event	Description	Applicable Variables	Event Category	QMGR Attribute
xmit_q_usage_error	<p>During a MQOPEN or MQPUT1 call a message was directed to a remote queue manager but one of the following occurred:</p> <ul style="list-style-type: none"> • 'Object queue manager name' specifies the name of a local queue, but it does not have a 'usage' attribute of 'transmission' • The 'object name' or 'object queue manager name' specifies the name of a local definition of a remote queue but one of the following applies to the 'xmit queue name' attribute definition: <ul style="list-style-type: none"> • 'Xmit queue name' is not blank, but specifies a queue that does not have a usage attribute of 'transmission' • 'Xmit queue name' is blank, but 'remote queue manager name' specifies a queue that does not have a usage attribute of 'transmission' 	<ul style="list-style-type: none"> • IMFQ_EVENT_QMGRNAME: Name of queue manager that generated event, whether local or remote • IMFQ_EVENT_QNAME: Name of queue for which caused the error • IMFQ_EVENT_APPLTYPE: Type of application that caused event. For example: MQAT_UNKNOWN MQAT_NO_CONTEXT MQAT_CICS MQAT_OS390 MQAT_IMS MQAT_OS2 MQAT_DOS MQAT_AIX MQAT_QMGR MQAT_OS400 MQAT_WINDOWS MQAT_CICS_VSE MQAT_WINDOWS_NT MQAT_VMS MQAT_GUARDIAN MQAT_VOS MQAT_IMS_BRIDGE MQAT_XCF MQAT_CICS_BRIDGE MQAT_NOTES_AGENT • IMFQ_EVENT_APPLNAME: Name of application • IMFQ_EVENT_OBJECTQMGRNAME: Name of the object queue manager • IMFQ_EVENT_XMITQNAME: Transmission queue name, not available for some events 	Queue Manager	remoteev

Variables for MQSeries Events

This section describes SHARED variables and variables that are supplied by MAINVIEW AutoOPERATOR and set by MQSeries events.

SHARED Variables

Table 5-3 contains descriptions for SHARED variables available to MAINVIEW AutoOPERATOR Rules and EXECs.

Table 5-3 SHARED Variables for MAINVIEW AutoOPERATOR for MQSeries

Shared Variable Name	Description
qaoqaost	Status of MAINVIEW AutoOPERATOR for MQSeries. Possible values are ACT or INACT.
QQMSxxxx or qqms.xxxx	Status of connection to queue manager where xxxx is the subsystem ID (SSID) of the queue manager. This variable is valid only for queue managers with a valid connection. The absence of this variable may mean the queue manager is not active, or no Rules are enabled which access this queue manager.
QCPFxxxx or qcpf.xxxx	Command prefix string (CPF) where xxxx is the SSID of the queue manager. This variable is available only if a connection has been established.
QDEDxxxx or qded.xxxx	Name of the dead letter queue where xxxx is the SSID of the queue manager. This variable is available only if a connection has been established.
QCMDxxxx or qcmd.xxxx	Name of the command input queue where xxxx is the SSID of the queue manager. This variable is available only if a connection has been established.

Variables Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages

Table 5-4 on page 5-56 contains descriptions of variables that are supplied by MAINVIEW AutoOPERATOR available to Rules and EXECs for all MQSeries messages. The following table describes the most commonly used variables that are available to a Rule or when an EXEC is scheduled from a MQSeries Rule or a combination of both. For other structures not listed here, for example DLH (Dead Letter Header) and others, please refer to Appendix A, “Diagnosing MAINVIEW AutoOPERATOR for MQSeries Errors.” Variables created from structures begin with “IMFQ_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore.

For definitions of field names and constants used in this table, which is derived from MQSeries structure field names, see the *MQSeries Application Programming Reference Guide*.

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages (Part 1 of 7)

Variable Name	Description
<p>Note: These variables follow the IBM conventions where possible. Therefore, the variables lists in this table may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>	
imfoqmgr	Subsystem ID of queue manager.
imfqcpf	OS/360 console command prefix for this MQ queue manager.
imfqrmt	Origin of message (Y: message is from a remote queue, N: message is not from a remote queue).
IMFQ_STRUCTURES	Contains a blank-delimited list of the MQSeries structures contained in the message. For example, in the case of a Dead Letter message, the contents would contain at least: MD DLH
IMFQ_OFFSETS	Contains blank-delimited list of offsets to structures and application data from the front of the message buffer. For example: if the message buffer contains a Dead Letter Header (DLH) and application data, the variable value may look like this: 'NONE 0 172' indicating NONE for MD (since it is not in the message buffer), 0 for the DLH (first structure in the buffer) and 172 for application data (follows the DLH).
IMFQ_QAO_CATCH_UP_MSG	Indicates whether the current message is a potential Catch-up message (present on the queue when MAINVIEW AutoOPERATOR opened it). Possible values are <ul style="list-style-type: none"> • Y (Yes): the message already existed on the queue • N (No): the message has newly arrived since the queue was opened
IMFQ_MD_STRUCID	Contains the value MQMD_STRUC_ID.
IMFQ_MD_VERSION	Contains either MQMD_VERSION_1 or MQMD_VERSION_2.

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages
(Part 2 of 7)

Variable Name	Description
IMFQ_MD_REPORT	Contains one or more of the following delimited by blanks: MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE <i>See note at the beginning of this table.</i>
IMFQ_MD_MSGTYPE	Contains one of the following values: MQMT_DATAGRAM MQMT_REQUEST MQMT_REPLY MQMT_REPORT <i>See note at the beginning of this table.</i>
IMFQ_MD_EXPIRY	Contains a decimal value in the range 1 to 999999999, or MQEI_UNLIMITED.

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages (Part 3 of 7)

Variable Name	Description
IMFQ_MD_FEEDBACK	<p>Contains one of the following values for report messages:</p> <p style="padding-left: 40px;">MQFB_NONE MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT</p> <p>In addition, this field can contain any reason code from the following sources:</p> <ul style="list-style-type: none"> • IMS-bridge feedback codes • CICS-bridge feedback codes • MQSeries reason codes <p>Note: The content of this field can be a feedback code or a reason code. You can determine whether it is a feedback code or reason code by its value. For example, feedback codes start with MQFB_xxxx and reason codes start with MQRC_xxxx.</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_ENCODING	<p>Contains either MQENC_NATIVE or any decimal value up to 999999999.</p>
IMFQ_MD_CODEDCHARSETID	<p>Contains</p> <p style="padding-left: 40px;">MQCCSI_Q_MGR MQCCSI_EMBEDDED</p> <p>or any decimal value up to 999999999.</p> <p>See IBM <i>MQSeries Application Programming Reference Guide</i> for details on how the CodedCharSetId field is used as input.</p>

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages (Part 4 of 7)

Variable Name	Description
IMFQ_MD_FORMAT	<p>Contains one of the following values:</p> <p>MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_PRIORITY	Contains a decimal value in the range 0 to 999999999.
IMFQ_MD_PERSISTENCE	<p>Contains one of the following values:</p> <ul style="list-style-type: none"> • MQPER_PERSISTENT • MQPER_NOT_PERSISTENT
IMFQ_MD_MSGID	Contains up to a 32-byte Msgid. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_CORRELID	Contains up to a 32-byte CorrelId. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_BACKOUTCOUNT	Contains a decimal value in the range 0 to 255.
IMFQ_MD_REPLYTOQ	Contains up to a 48-character name of the ReplyToQ.
IMFQ_MD_REPLYTOQMGR	Contains up to a 48-character name of the ReplyToQMGR.
IMFQ_MD_USERIDENTIFIER	Contains up to a 12-character UserIdentifier.
IMFQ_MD_ACCOUNTINGTOKEN	Contains either MQACT_NONE or up to a 32-byte AccountingToken. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_APPLIDENTITYDATA	Contains up to a 32-character value for ApplIdentityData.

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages (Part 5 of 7)

Variable Name	Description
IMFQ_MD_PUTAPPLTYPE	Contains a user-defined type or one or more of the following standard types: MQAT_AIX MQAT_BROKER MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DEFAULT MQAT_DOS MQAT_DQM MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_JAVA MQAT_NO_CONTEXT MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <i>See note at the beginning of this table.</i>
IMFQ_MD_PUTAPPLNAME	Contains up to a 28-character value for PutAppName.
IMFQ_MD_PUTDATE	Contains an 8-character date stamp.
IMFQ_MD_PUTTIME	Contains an 8-character time stamp.
IMFQ_MD_APPLORIGINDATA	Contains a 4-character value for ApplOriginData.
IMFQ_MD_GROUPID	If a MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GroupId. Processed exactly as received. No conversion of hexadecimal data is done.
IMFQ_MD_MSGSEQNUMBER	If a MQMD_VERSION_2 MD is present, this variable may contain a decimal value in the range 1 to 999999999.
IMFQ_MD_OFFSET	If a MQMD_VERSION_2 MD is present, this variable may contain a decimal value in the range 0 to 999999999.

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages (Part 6 of 7)

Variable Name	Description
IMFQ_MD_MSGFLAGS	<p>If a MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks:</p> <p style="text-align: center;">MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MD_ORIGINALLENGTH	<p>Contains a decimal value in the range 1 to 999999999, or MQOL_UNDEFINED.</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_QATTR_BACKOUTREQUEUEQNAME	Contains up to a 48-character value.
IMFQ_QATTR_BACKOUTTHRESHOLD	Contains a decimal number in the range 0 to 999999999.
IMFQ_QATTR_CREATIONDATE	Contains a 12-character date.
IMFQ_QATTR_CREATIONTIME	Contains an 8-character time.
IMFQ_QATTR_QDESC	Contains up to a 48-character value.
IMFQ_QATTR_CURRENTQDEPTH	Contains a decimal number for the current queue depth.
IMFQ_QATTR_DEFPRIORITY	Contains a decimal number in the range 0 to the MaxPriority of the queue manager.
IMFQ_QATTR_DEFPERSISTENCE	<p>Contains one of the following:</p> <p style="text-align: center;">MQPER_PERSISTENT MQPER_NOT_PERSISTENT</p>
IMFQ_QATTR_DEFINPUTOPENOPTION	<p>Contains one of the following:</p> <p style="text-align: center;">MQOO_INPUT_EXCLUSIVE MQOO_INPUT_SHARED</p>
IMFQ_QATTR_DEFINITIONTYPE	<p>Contains one of the following:</p> <p style="text-align: center;">MQQDT_PREDEFINED MQQDT_PERMANENT_DYNAMIC MQQDT_TEMPORARY_DYNAMIC MQQDT_SHARED_DYNAMIC</p>
IMFQ_QATTR_INHIBITGET	<p>Contains one of the following:</p> <p style="text-align: center;">MQQA_GET_ALLOWED MQQA_GET_INHIBITED</p>
IMFQ_QATTR_HARDENGETBACKOUT	<p>Contains one of the following:</p> <p style="text-align: center;">MQQA_BACKOUT_HARDENED MQQA_BACKOUT_NOT_HARDENED</p>
IMFQ_QATTR_INITIATIONQNAME	Contains up to a 48-character value.

Table 5-4 Variables That Are Supplied by MAINVIEW AutoOPERATOR for All MQSeries Messages (Part 7 of 7)

Variable Name	Description
IMFQ_QATTR_OPENINPUTCOUNT	Contains a decimal value.
IMFQ_QATTR_MAXQDEPTH	Contains a decimal number in the range 0 to 999999999.
IMFQ_QATTR_MAXMSGLENGTH	Contains a decimal number in the range 0 to 104857600.
IMFQ_QATTR_MSGDELIVERYSEQUENCE	Contains one of the following: MQMDS_FIFO MQMDS_PRIORITY
IMFQ_QATTR_OPENOUTPUTCOUNT	Contains a decimal value.
IMFQ_QATTR_PROCESSNAME	Contains up to a 48-character value.
IMFQ_QATTR_INHIBITPUT	Contains one of the following: MQQA_PUT_INHIBITED MQQA_PUT_ALLOWED
IMFQ_QATTR_RETENTIONINTERVAL	Contains a decimal number in the range 0 to 999999999.
IMFQ_QATTR_SHAREABILITY	Contains one of the following: MQQA_SHAREABLE MQQA_NOT_SHAREABLE
IMFQ_QATTR_STORAGECLASS	Contains up to a 8-character value.
IMFQ_QATTR_TRIGGERDAT	Contains up to a 64-character value
IMFQ_QATTR_TRIGGERDEPTH	Contains a decimal value.
IMFQ_QATTR_TRIGGERCONTROL	Contains one of the following: MQTC_OFF MQTC_ON
IMFQ_QATTR_TRIGGERMSGPRIORITY	Contains a decimal value in the range of 0 to the maximum priority.
IMFQ_QATTR_TRIGGERTYPE	Contains one of the following: MQTT_NONE MQTT_FIRST MQTT_EVERY MQTT_DEPTH
IMFQ_QATTR_USAGE	Contains one of the following: MQUS_NORMAL MQUS_TRANSMISSION

MQS Selection Criteria and Action Specification Panel Fields

This section describes the fields for the panels introduced with MAINVIEW AutoOPERATOR for MQSeries: Selection Criteria - MQS and Action Specification - MQS. Use these panels to specify the selection criteria and the actions to perform with each Rule.

Selection Criteria for MQS Panel

The Selection Criteria - MQS panel is used for specifying selection criteria, as shown in Figure 5-28. All criteria must be met before a Rule's actions will be taken.

Figure 5-28 Selection Criteria - MQS Field Description

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>>>                                     TGT --- SYSE

                Rule-set === AAORUL00                Rule-id === MQCICS

Queue Identification:                                (1 to 12 Queue Managers)
Manager(s)    ==>> CSQ1
Queue Id      ==>> CICSPRD1.WAREHOUS.INVENTORY

Message Identification:
Format        ==>> MQFMT_CICS                        (Value from MD Format field)
Event Type    ==>>                                     (Enter ? for help)

                Sub  Len  Op  Value
Msgid         ==>> 10_ : 2__ HE  A3C4_____
CorrelId      ==>> 20_ : 1__ HG   1_____
Msg Buffer     ==>> 40_ : 12_ EQ  ITEM#3445_____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes

```

Use the Selection Criteria - MQS panel to specify attributes of an event that the Rule must match before it will fire. Table 5-5 on page 5-64 lists the Selection Criteria - MQS panel's field names and a description of each.

Table 5-5 Selection Criteria - MQS Panel's Field Names (Part 1 of 4)

Field Name	Description
Manager(s)	<p>Type a four-character queue manager name in this field. You can specify up to 12 managers, each separated by either blanks or commas.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> • Pattern matching is permitted in this field; <i>variables are not valid</i>. • An entry in this field is required. • There is no default for this field.
Queue ID	<p>Type 1 to 48 characters for the name of a specific MQSeries queue. If not specified, the source of the message can be any queue on which MAINVIEW AutoOPERATOR is listening for messages.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> • Local and Shared variables are permitted. Pattern matching is also permitted. • An entry in this field is not required. • This field is left blank by default. • The specified queue must have an entry in BBPARM member AAOMQL00 for the Rule to monitor the queue.
Format	<p>Type the message format in this field. Use to select messages based on the format found in the message descriptor. The following formats are recognized by MAINVIEW AutoOPERATOR:</p> <p>MQFMT_EVENT: Event message USER: User defined format, any message format except MQFMT_EVENT; see the following list.</p> <p style="padding-left: 40px;">MQFMT_NONE: No format name MQFMT_ADMIN: Command server request/reply message MQFMT_CHANNEL_COMPLETED: Channel completed 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_IMS: IMS information header MQFMT_IMS_VAR_STRING: IMS variable string MQFMT_MD_EXTENSION: Message-descriptor extension MQFMT_PCF: User-defined msg in programmable command format MQFMT_REF_MSG_HEADER: Reference message header MQFMT_STRING: Message consisting entirely of characters MQFMT_TRIGGER: Trigger message MQFMT_WORK_INFO_HEADER: Work information header MQFMT_XMIT_Q_HEADER: Transmission queue header MQFMT_RF_HEADER: Rules and formatting header MQFMT_RF_HEADER_2: Rules and formatting header version 2</p> <p>This field has no default.</p>

Table 5-5 Selection Criteria - MQS Panel's Field Names (Part 2 of 4)

Field Name	Description
Event Type	<p>Use 1 to 28 characters to define a particular event type for which messages will be selected when the message format is MQFMT_EVENT. For a list of valid values, type a '?'.</p> <p>Use only if the Format field is set to MQFMT_EVENT.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> • An entry in this field is not required • This field is left blank by default to fire on all event types
Msgid	<p>Use this field to select messages containing a specific Msgid in the Message Descriptor Structure. Also, you can use the following input fields to specify a substring, a length and a special operator to use in comparisons:</p> <p>Sub (Substring): Specifies the starting point of the comparison. Optional. Valid values are 1 to 24.</p> <p>Len (Length): Specifies the length of the substring comparison. Optional. If omitted, length is the remainder of the field following the substring. Valid values are 0 to 24.</p> <p>Op (Operator): Specifies the type of data and the type of comparison. Valid operators are EQ, NE, GT, LT, GE, LE, IN, EX, HE, HN, HG, HL, GX, LX, FO and FN. The default is EQ. To view a description of the operators, see the Help panel for this field.</p> <p>Value (Value): Specifies the data value against which the Msgid field of the Message Descriptor Block is compared. The data type must match that which is indicated by the Op field. Local and Shared variables are permitted.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> • You can use this field only for messages that are not of format, MQFMT_EVENT • An entry in this field is not required. • This field is left blank by default.

Table 5-5 Selection Criteria - MQS Panel's Field Names (Part 3 of 4)

Field Name	Description
Correlld	<p>Use this field to select messages containing a specific Correlld in the Message Descriptor Structure. Also, you can use the following input fields to specify a substring, a length and a special operator to use in comparisons:</p> <p>Sub (Substring): Specifies the starting point of the comparison. Optional. Valid values are 1 to 24.</p> <p>Len (Length): Specifies the length of the substring comparison. Optional. If omitted, length is the remainder of the field following the substring.</p> <p>Op (Operator): Specifies the type of data and the type of comparison. Valid operators are EQ, NE, GT, LT, GE, LE, IN, EX, HE, HN, HG, HL, GX, LX, FO and FN. Default is EQ. To view a description of the operators, see the Help panel for this field.</p> <p>Value (Value): Specifies the data value against which the Msgid field of the Message Descriptor Block is compared. The data type should match that indicated by the Op field. Local and Shared variables are permitted.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> • An entry in this field is not required. • Allowed only for messages that are not of format MQFMT_EVENT. • This field is left blank by default.

Table 5-5 Selection Criteria - MQS Panel's Field Names (Part 4 of 4)

Field Name	Description
Msg Buffer	<p>(Application Data) Use this field to select all messages that contain a specified text string. The first 255 bytes of the message participate in the Text String matching. You can use the following input:</p> <p>Sub (Substring): Specifies the starting point for a comparison to take place. Optional. Valid values are 1 to 255.</p> <p>Len (Length): Specifies the length of the substring comparison. Optional. If omitted, length is the remainder of the field following the substring. Valid values are 0 to 255.</p> <p>Op (Operator): Specifies the type of data and the type of comparison. Valid operators are EQ, NE, GT, LT, GE, LE, IN, EX, HE, HN, HG, HL, GX, LX, FO and FN. Default is EQ. To view a description of the operators, see the Help panel for this field.</p> <p>Value (Value): Specifies the data value against which the data from the Msg Buffer is compared. The data type should match that indicated by the Op field. Local and Shared variables are permitted.</p> <p>Note: Be aware that the variable, IMFTEXT, which is passed to an EXEC when scheduled from a Rule, has the same limitations as the MSG Buffer of the Selection Criteria Panel, 255 bytes. For more information regarding IMFTEXT, see Chapter 4 in the <i>MAINVIEW AutoOPERATOR Advanced Automation Guide</i>. To access data in the message beyond 255 bytes, you must use an EXEC to GET the message from the queue. Using the IMFEXEC MQI GET statement in an EXEC, you can access up to 32767 bytes of the message content.</p> <p>Other notes about this field:</p> <ul style="list-style-type: none"> • An entry in this field is not required. • Allowed only for messages that are not of format MQFMT_EVENT. • This field is left blank by default.

Action Specification for MQS Panel

The Action Specification - MQS panel, shown in Figure 5-29, is one of the two panels used for specifying an action when a Rule fires; ALERT Action(s) is the other.

Additionally, the Action Specification - MQS panel can be used to remove the Dead Letter Header and as a Destination Queue Manager field for use with the Copy function.

Figure 5-29 Action Specification - MQS Field Description

```

BMC Software ----- Action Specification - MQS ----- AutoOPERATOR
COMMAND ==> TGT --- SYSE

      Rule-set === AAORUL00          Rule-id === DLH
Automation Actions:
Journal          ==>
EXEC Name/Parms  ==>
Cmd (Type MQ )  ==>

Set Variable     ==>                ==>
Notify           ==>                Outboard Pager ID
Info             ==>

DOM Id           ==>                Delete Operator Message
Issue WTO Msg    ==>

MQSeries Automation Actions:
Keep Message     ==> YES (Yes or NO)      Remove DLH ==> (Yes or No)
Destination Que  ==>
Destination QMgr ==>

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

Table 5-6 lists the Action Specification - MQS panel’s field names and a description of each.

Table 5-6 Action Specification - MQS Panel’s Field Names (Part 1 of 4)

Field Name	Description														
EXEC Name/Parms	This field allows up to 56 characters and is used to schedule an EXEC to perform advanced automation that cannot be performed by a Rule. Local and SHARED variables are valid. The EXEC parameters may be specified, which are passed when the EXEC is scheduled for execution. <ul style="list-style-type: none"> • An entry in this field is not required. • This field is left blank by default. 														
Cmd(Type)	This field allows up to 126 characters and is used to issue one or more console commands. Local and SHARED variables are valid. The following is a list of command types and origin: <table border="0" style="width: 100%;"> <tr> <td>Blank</td> <td>No command is issued.</td> </tr> <tr> <td>MVS</td> <td>MVS command is issued from event originating address space.</td> </tr> <tr> <td>BBI</td> <td>BBI command is issued from the BBI-SS PAS address space.</td> </tr> <tr> <td>IMS</td> <td>IMS command is issued from the BBI-SS PAS address space.</td> </tr> <tr> <td>CICS</td> <td>CICS command is issued from the BBI-SS PAS address space.</td> </tr> <tr> <td>MQ</td> <td>MQ command is issued from the BBI-SS PAS address space.</td> </tr> <tr> <td>SS</td> <td>MVS command is issued from the BBI-SS PAS address space.</td> </tr> </table>	Blank	No command is issued.	MVS	MVS command is issued from event originating address space.	BBI	BBI command is issued from the BBI-SS PAS address space.	IMS	IMS command is issued from the BBI-SS PAS address space.	CICS	CICS command is issued from the BBI-SS PAS address space.	MQ	MQ command is issued from the BBI-SS PAS address space.	SS	MVS command is issued from the BBI-SS PAS address space.
Blank	No command is issued.														
MVS	MVS command is issued from event originating address space.														
BBI	BBI command is issued from the BBI-SS PAS address space.														
IMS	IMS command is issued from the BBI-SS PAS address space.														
CICS	CICS command is issued from the BBI-SS PAS address space.														
MQ	MQ command is issued from the BBI-SS PAS address space.														
SS	MVS command is issued from the BBI-SS PAS address space.														

Table 5-6 Action Specification - MQS Panel's Field Names (Part 2 of 4)

Field Name	Description
Cmd(Type) continued	<p>Note: Multiple commands are delimited by two colons. CICS commands must contain a target. For example: target:commandtext If the Rule event type is not MQS or the targeted local-system OS/390 queue manager is not the queue manager firing the Rule, MQ commands must begin with the name of the queue manager processing the specified command. For example: queuemgrname: command text Optional keywords for command type MQ:</p> <ul style="list-style-type: none"> • LM: Name of local MQSeries queue manager. This keyword is necessary only when issuing a command to a remote queue manager connected to an OS/390 queue manager that is not the current queue manager firing the Rule. The keyword can be abbreviated to L. • QM: Name of the remote MQSeries queue manager to which the command should be sent. This is an optional keyword and can be abbreviated to Q. • PCF: Specifies command should be issued in Programmable Command Format. This is an optional keyword (default is N). Most non-OS/390 platforms require commands in PCF format (Y). Can be abbreviated to P. • CQ: Name of command queue. Optional keyword used in conjunction with QM parameter. Default is SYSTEM.ADMIN.COMMAND.QUEUE. Can be abbreviated to C. If QM is not specified (or is equal to LM), the CQ keyword will be ignored. <p>Example 1: Issue ALTER QMGR on local MQSeries queue manager MQ03. CMD TYPE(MQ) ==> MQ03:ALTER QMGR INHIBTEV(ENABLED)</p> <p>Example 2: Issue ALTER QMGR on remote queue manager through local queue manager MQ03, where MQ03 is the queue manager on which the PUT was issued that caused the Rule to fire. The local queue manager has a connection with the remote queue manager. CMD TYPE(MQ) ==> QM(WINT.TE01) P(Y): ALT QMGR INHIBTEV(ENABLED)</p> <p>Example 3: Issue ALTER QMGR on remote queue manager through local queue manager MQ04, even though the queue manager on which the PUT was issued is MQ03. CMD TYPE(MQ) ==> LM(MQ04) QM(WINT.TE01) P(Y): ALT QMGR INHIBTEV(ENABLED)</p>
Set Variable	<p>In the first entry, up to 33 characters are allowed. In this field specify the name of the variable to be modified. Only SHARED variables will be created or updated. SHARED and local variables and constants may be used to create the variable name. In the second entry, up to 17 characters are allowed. SHARED and local variables and constants are valid. Use this field to assign a new value (using a variable or unsigned constant) or modify (using a signed constant).</p> <ul style="list-style-type: none"> • An entry in this field is not required. • This field is left blank by default.
Notify	<p>This field allows up to 33 characters and is used to store the telephone number or MAINVIEW AutoOPERATOR Elan Operator ID to be notified using an outboard pager.</p> <ul style="list-style-type: none"> • Specify the telephone number exactly as you would dial it on the telephone. • The Elan Operator ID must exist in the Elan Operator Profile. <p>This field is used in conjunction with the Info ==> field.</p> <ul style="list-style-type: none"> • Local and SHARED variables are valid. • An entry in this field is not required. • This field is left blank by default.

Table 5-6 Action Specification - MQS Panel's Field Names (Part 3 of 4)

Field Name	Description
Info	<p>This field allows up to 80 characters and is used to specify the information to appear in the display area of an outboard pager managed by MAINVIEW AutoOPERATOR Elan.</p> <ul style="list-style-type: none"> Local and SHARED variables are valid. An entry in this field is not required. This field is left blank by default.
DOM ID	<p>This field allows up to 33 characters and is used to allow a Rule to issue a Delete Operator Message (DOM) to delete a highlighted message from the operator console.</p> <ul style="list-style-type: none"> SHARED variables are valid. An entry in this field is not required. This field is left blank by default.
Issue WTO Msg	<p>This field allows up to 125 characters and allows a Rule to issue a message to a console (Write To Operator). Specify the message you would like issued to the operator console. The message is issued without routing or descriptor codes. The message may be selected by another Rule.</p> <ul style="list-style-type: none"> An entry in this field is not required. This field is left blank by default.
Keep Message	<p>You can enter Yes or No to specify whether or not to keep the message in the source MQSeries queue.</p> <p>Yes Do not delete the message from the source MQSeries queue.</p> <p>No Delete the message from the source MQSeries queue.</p> <ul style="list-style-type: none"> An entry in this field is not required. The default is Yes.
Destination Queue	<p>You can enter up to 48 characters in this field for the name of the MQSeries queue to copy/move messages to.</p> <ul style="list-style-type: none"> Local and SHARED variables are valid. An entry in this field is not required. This field is left blank by default. <p>Using "Keep Message=NO and Destination Queue" causes a message to be moved from one queue to another while "Keep Messages=YES and Destination Queue" causes a message to be copied from one queue to another.</p> <p>If Keep(YES) and destination queue are both specified, the Keep/Copy will be synchronized.</p>

Table 5-6 Action Specification - MQS Panel's Field Names (Part 4 of 4)

Field Name	Description
Remove DLH	<p>During a Copy or Move operation, if YES is specified in this field, the original dead letter message is rebuilt by removing the Dead Letter Header and restoring the original Encoding, CodedCharSetId and Format fields back into the message descriptor.</p> <p>Note: After removing the DLH, any Rules following this Rule will not see the DLH but will see the rebuilt message.</p> <p>Possible values are Yes No</p> <p>Using this function, you can strip a dead letter message of all information regarding dead letter processing and forward it to an alternate queue for normal processing. In addition, when using this option, a message passed to a Rule-scheduled EXEC will not contain the Dead Letter Header.</p>
Destination Mgr	<p>A queue manager to which the message is targeted during a Copy or Move operation can be specified in this field. Without this specification, the message is PUT to a destination queue on the current queue manager for which the Rule fired. To use this field, the Destination Queue field must be set. The specified queue manager ID is placed into the object descriptor so that normal MQSeries queue and queue manager resolution take place.</p>

Tracking Automation Statistics for MQS Events

The *MAINVIEW AutoOperator Basic Automation Guide* describes how to activate the Automation Reporter application, what it does, and how you can read the data collected by the application. For MAINVIEW AutoOPERATOR for MQSeries, the data collected by the Automation Reporter can be seen in the record types EVNT and ACTN.

The Automation Reporter record for event type EVNT tracks the number of times the event type MQS appears in the Event Statistics Table. An example of the record can be seen in Figure 5-30.

Figure 5-30 Example of EVNT Automation Reporter Record

```

SYSC,EP01,310,BBPLEX01,"19970210","09:47",13,EVNT,ONLY,"19970210",
"10:00","PUT_INHIBITED",15,15,MQS,"",4,EOR
SYSC,EP01,310,BBPLEX01,"19970210","09:47",13,EVNT,ONLY,"19970210",
"10:00","CHANNEL_STOPPED",1,1,MQS,"",4,EOR

```

The Automation Reporter record for event type ACTN tracks the number of times automation handled events and what actions were taken. This record includes tracking of MQS events. An example of the record can be seen in Figure 5-31.

Figure 5-31 Example of ACTN Automation Reporter Record

```
SYSC,EP01,310,BBPLEX01,"19970210","10:00",10,ACTN,ONLY,"19970210",  
"10:00",13,7,MSG,67,0,0,0,0,0,CMD,9,0,0,0,0,0,JRNL,3,0,0,0,0,0,  
TIME,0,0,0,0,0,0,ALRT,0,0,0,0,0,0,EXT,0,0,0,0,0,0,IMS,0,0,0,0,0,0,  
CICS,0,0,0,0,0,0,DB2,0,0,0,0,0,0,JES3,0,0,0,0,0,0,  
ALRM,54,0,0,0,0,0,VAR,1,0,0,0,0,0,MQS,1,0,0,0,0,0,EOR
```

For a complete description of the data shown here, refer to the *MAINVIEW AutoOPERATOR Basic Automation Guide*.

Chapter 6 Using MAINVIEW AutoOPERATOR for MQSeries Solutions

This chapter describes how to use the following MAINVIEW AutoOPERATOR for MQSeries solutions:

- implementation considerations
- Dead Letter Queue Management solution
- System Queue Archival solution
- Service Interval Performance solution
- Queue Depth Management solution
- Channel Availability solution
- Basic Intercommunication solution

Implementation Considerations

The MAINVIEW AutoOPERATOR solutions are primarily Rule-based and, therefore, easy to implement.

However, if your site does not have the IBM library for SAA REXX/370 installed and you plan to run the System Queue Archival solution, the Dead Letter Queue solution, or the Channel Availability solution, you must review the chapter titled “Activating the REXX/370 Alternate Library” in the *MAINVIEW AutoOPERATOR Customization Guide*. These solutions require that either the SAA REXX/370 library or the REXX370/ alternate library be installed.

All of the Rules for these solutions are in Rule Sets AAORULBQ, AAORULBH, and AAORULBR. You must edit and enable the appropriate Rules before the solutions are usable.

In addition, ensure that the automation strategy for Rules is set to INDIVIDUAL and the Rule Sets AAORULBQ, AAORULBH, and AAORULBR each have a strategy set to ALL. For more information about how to set automation strategy for MAINVIEW AutoOPERATOR Rules and Rule Sets, refer to the *MAINVIEW AutoOPERATOR Basic Automation Guide*.

System Queue Archival

The System Queue Archival solution uses an EXEC named QMQUNLDQ. This EXEC writes to the SYSIN file to pass control statements to IBM MQSeries utility programs. To provide serialization to the SYSIN file, QMQUNLDQ uses the IMFEXEC VENQ service. The minor name that QMQUNLDQ enqueues on is ARCSYSIN. Therefore, BMC Software strongly recommends that any user-written EXEC that uses the SYSIN file also uses the same technique. Any EXEC that uses the SYSIN file and does not use VENQ serialization may create unpredictable results.

Using the Dead Letter Queue Solution

The Dead Letter Queue solution performs different functions that can be invoked separately or together. Use the Dead Letter Management portion of the solution when you want to

- be informed automatically by an ALERT when a valid dead letter (a message with a dead letter header) arrives in the dead letter queue
- have invalid dead letters (messages without a dead letter header) moved from the dead letter queue to an alternate queue and have an ALERT created after the messages are moved

Use the Dead Letter Aging Management portion of the solution when you want to fire an EXEC once a day that generates two ALERTs:

- a nonvolatile ALERT containing the total number of messages that are more than one day old
- a nonvolatile ALERT containing the total number of messages that are more than one week old from the dead letter queue

The following table describes the different Rules that you can implement to perform various features of the Dead Letter Queue solution.

Rule ID	EXEC Scheduled	Triggering Event
MQDEDQ01	QMDEDQ1	Fires when an invalid dead letter arrives in the queue
MQDEDQ02	QMDEDQ1	Creates nonvolatile ALERTs which report on the total number of messages that are more than 1 day old and the total number of messages that are more than one week old in both the dead letter queue and the alternate queue for invalid dead letters
MQDEDQ03		Fires when a valid dead letter arrives in the queue

Security Considerations

To use the Dead Letter Queue solution, MAINVIEW AutoOPERATOR needs access authority to MQSeries resources as follows:

- Alter access to create the queue used to hold invalid dead letters and update access to put messages onto it. Specify the queue name when executing QMDEDQ1.
- Update authority for the dead letter queue on each queue manager that is automated by MAINVIEW AutoOPERATOR.

Implementing the Dead Letter Queue Solution

To invoke any of the Rules for the Dead Letter Queue solution, follow these procedures:

1. Enter **INCL** to add the dead letter queue names to BBPARM member AAOMQL_{xx}.
2. Issue the **RESET MQ xx** command where *xx* is the two-character suffix of the AAOMQL_{xx} member.

The following sections describe in greater detail how to enable specific Rules that make up the Dead Letter Queue solution. To activate the complete solution, you must enable all three Rules as described in these sections.

Enabling the Dead Letter Management Rules

The Dead Letter Management portion of the solution covers managing two different kinds of dead letters: dead letters, which are valid dead letters and have dead letter headers, and invalid dead letters, which do not have proper dead letter headers. The instructions in this section describe how to enable the Rules for both kinds of dead letters.

Invalid Dead Letter Management

To invoke the portion of the solution which manages invalid dead letters, follow these steps:

Step 1 Schedule an EXEC called QMQDEDQ1.

This EXEC creates a queue to which invalid dead letters (dead letters without proper headers) will be moved. You must pass the name of the queue manager ID as the first parameter and pass the name of the queue (to which invalid dead letters will be written). You can choose the queue name but the queue's attributes are modeled after the actual system dead letter queue.

To schedule the QMQDEDQ1 EXEC:

- 1.A** Type the following command from the BBI terminal session log display:

```
%QMQDEDQ1 queuemgrID queue.name
```

where *queuemgrID* is the queue manager ID and *queue.name* is the new queue name.

- 1.B** You must schedule this EXEC for each queue manager that this solution will manage, and you must use the same queue name so that the solution can manage all the queues.

Step 2 Edit and enable the Rule MQDEDQ01 in the AAORULBQ Rule Set.

When enabled, the MQDEDQ01 Rule fires when an invalid dead letter arrives and notifies the operator with an ALERT that the letter has been moved to the queue created by EXEC QMQDEDQ1.

To enable and edit the MQDEDQ01 Rule:

- 2.A** Go to the Automation Control panel where the list of Rule Sets is displayed.

- 2.B** Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
- 2.C** Select the AAORULBQ Rule Set.
- 2.D** Select the Rule MQDEDQ01.
- 2.E** Edit MQDEDQ01 using the Rules Processor application panels.
- 2.F** Press **Enter** to navigate through the panels and make the following changes:
 - On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command `.RESET MQ xx`.
 - On the Action Specification panel, change the Destination Queue from XXX.XXX.XXX to the name of the queue created for invalid dead letters.
 - On the Alert Action 1 panel, change XXX.XXX.XXX in the ALERT text to the name of the queue created for invalid dead letters.

Step 3 Enable MQDEDQ01 by performing *one* of the following actions:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E)nable line command on the Rule Set Overview panel.
- Issue the BBI command

.T RULE,ENA,MQDEDQ01

Valid Dead Letter Management

To invoke the portion of the solution which manages valid dead letters, you must enable the MQDEDQ03 Rule which fires every time a valid dead letter arrives in the queue and issues an ALERT to notify the operator that a valid dead letter has arrived.

To enable and edit the MQDEDQ03 Rule, follow these steps:

- Step 1** Follow Step 2.A on page 6-4 through Step 2.C on page 6-5.
- Step 2** Select the Rule MQDEDQ03.
- Step 3** Edit MQDEDQ03 using the Rules Processor application panels.

Press **Enter** to navigate through the panels and make the following change:

On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command .RESET MQ xx.

- Step 4** Enable MQDEDQ03 by performing *one* of the following actions:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E)nable line command on the Ruleset Overview panel.
- Issue the BBI command

.T RULE,ENA,MQDEDQ03

Enabling the Dead Letter Aging Management Rule

To invoke the Dead Letter Aging Management portion of the solution, you must edit and enable the MQDEDQ02 Rule in the AAORULBQ Rule Set.

The MQDEDQ02 Rule creates nonvolatile ALERTs which report on the total number of messages that are more than one day old and the total number of messages that are more than one week old in both the dead letter queue and the alternate queue for invalid dead letters (dead letters without proper headers). Note that nonvolatile ALERTs are written to DASD and can therefore survive a BBI-SS PAS cold start or even an OS/390 IPL.

To enable and edit the MQDEDQ02 Rule, follow these steps:

- Step 1** Go to the Automation Control panel where the list of Rule Sets is displayed.
- Step 2** Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
- Step 3** Select the AAORULBQ Rule Set.

Step 4 Select the Rule MQDEDQ02.

Step 5 Edit MQDEDQ02 using the Rules Processor application panels.

Step 6 Press **Enter** to navigate through the panels and make the following change:

On the Selection Criteria panel specify what time of day you want the Rule to fire and set the interval to **24:00:00**.

Step 7 Enable the MQDEDQ02 Rule by performing *one* of the following actions:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the **(E)**nable line command on the Ruleset Overview panel.
- Issue the BBI command

.T RULE,ENA,MQDEDQ02

Stopping the Dead Letter Queue Solution

To inactivate any part of the solution, disable the Rules by performing *one* of the following actions:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the **(D)**isable line command on the Ruleset Overview panel.
- Issue the BBI command

.T RULE,DIS,ruleID

where *ruleID* is the Rule ID of the Rule you want to disable.

Using the System Queue Archival Solution

Use the System Queue Archival solution to move messages from a system instrumentation event queue (or event queue) that has reached its capacity. The messages for that queue are off-loaded automatically to a sequential generation data group (GDG) data set.

The Archival solution uses a Rule that is fired when an event queue fills to a user-specified percentage. The messages within the queue are then off-loaded by an MQSeries offload utility and moved to an OS/390 generation data set—thereby freeing the queue to receive more messages.

Rule ID	Default State	EXEC Scheduled	Triggering Event	Default Value
MQARC001	Disabled	QMQUINLDQ	Fires when an event queue fills to a user-specified percentage. Calls EXEC QMQUINLQ2 to set parameters used for the definition of the temporary queue that holds the messages prior to unloading them with CSQUTIL.	N/A
MQARC002 - MQARC005	Disabled	None	Fires when MAINVIEW AutoOPERATOR connects to the queue manager. These Rules Set variables are used by the solution during archival of the messages.	
MQARC002	Disabled	None	Specifies the name of the GDG to be used as archive.	No default value
MQARC003	Disabled	None	Causes an MQSeries offload utility to move the messages to the specified SYSOUT class.	*
MQARC004	Disabled	None	Specifies the device unit to which a generation data set is allocated.	SYSDA
MQARC005	Disabled	None	Specifies the block size with which a generation data set is allocated.	0
MQARC006	Disabled	None	Specifies the number of tracks with which a generation data set is allocated.	(190,19)

Security Considerations

To use the System Queue Archival solution, MAINVIEW AutoOPERATOR needs access authority to MQSeries resources as follows:

- Read access to the SYSTEM.ADMIN.QMGR.EVENT queue.
- Read access to the SYSTEM.ADMIN.PERFM.EVENT queue.
- Read access to the SYSTEM.ADMIN.CHANNEL.EVENT queue.

- Read access to the SYSBMC.ssid.EVENTS queue (if running Event Listener).
- Read access to the SYSBMC.DISTRIB.EVENTS queue (if running Event Listener).
- Alter access to create and delete the work queue used for copying the contents of the event queues before unloading to the GDG data set. Depending on whether you allow the default BMC-defined queue to be used or choose (using EXEC QMQUNLQ2) to select a queue name of your choice, you will need access as follows:
 - default BMC-defined queue name

 BBOMVAO.SYSTEM.ADMIN.*.EVENT.D*.T*

 where the first * is QMGR, PERFM or CHANNEL, the second * is the Julian date, and the third * is the time (in the format HHMM).
 - user's choice

 Whatever you specify for QLOCAL(queue.name) in QMQUNLQ2.

Implementing the System Queue Archival Solution

To invoke the System Queue Archival solution, follow these steps:

- Step 1** Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx.
- Step 2** Issue the .RESET MQ xx command where xx is the two-character suffix of the AAOMQLxx member.
- By default, the solution uses the attributes of the event queue being archived, along with the queue name BBOMVAO.SYSTEM.ADMIN.*.EVENT.D*.T* to define the BMC-defined temporary queue name used for copying contents of the event queues.
- Step 3** If the queue name is not acceptable or the attributes of the archived queue are not acceptable attributes for the temporary queue, copy BBPROC member QMQUNLQ2 to your own Exec library ahead of the BBPROC data set in your SYSPROC concatenation and specify your choice of queue name and/or queue attributes. Be sure to:
- Update the UNLDQ.0 variable with the number of variables you plan to create.

- Specify only the necessary variables and attributes because a maximum of 250 characters is allowed in the command string.
- Realize that it is not necessary to update this EXEC. If the EXEC is not updated, by default, the necessary variables for QMQUNLDQ to use to define the temporary queue are already set.

Enabling the System Queue Archival Rules

To invoke the System Queue Archival solution, perform the following steps:

- Step 1** Edit and execute JCL in BBSAMP member QMARCDEF to create the Generation Data Group (GDG) Base.

Options are available for where you can archive event queues. For example, you might want the QMGR events to be archived separately from the PERFM and CHANNEL events. If you are using the BMC Software Event Listener program, all events for a queue manager are collected in one queue specified by BMC Software. However, you might want them to be archived separately by queue manager. By default, all event queues for all queue managers monitored by one MAINVIEW AutoOPERATOR PAS are archived into one Generation Data Group structure.

The following options for defining and using the GDG data sets are available:

- First option: One GDG group for all Event queues and all queue managers
 - A. Decide what name you want to give your GDG data set.
 - B. Specify that name in the QMARCDEF job stream and in the Rule (as instructed in the steps that follow).
 - C. Specify **Unique(N)** as an input parameter to EXEC QMQUNLDQ in Rule MQARC001. This is the default setting.
 - D. Using this option, if you chose SYS1.MQARC for your GDG base data set name, specify this value in rule MQARC002 (as stated in the following instructions) and in the data set specification in job QMARCDEF in BBSAMP. The resulting GDG data set names would be SYS1.MQARC.G0001V00 through SYS1.MQARC.G0255V00, incrementing by one each time the solution is invoked.

- Second option: One GDG group per queue manager for all event queues when the BMC Software MQSeries Event Listener product is running
 - A. Decide what name you want to give your GDG data set.
 - B. Add the queue manager and partial name 'SYSBMC', separated by periods, to the data set name.
 - C. Specify **Unique(Y)** as an input parameter to EXEC QMQUNLDQ in Rule MQARC001.
 - D. Using this option, if you chose SYS1.MQARC for your GDG base data set name and your queue manager ID is CSQ1, you would add .CSQ1.SYSBMC to the base name in the data set specification in job QMARCDEF in BBSAMP. In Rule MQARC002, specify SYS1.MQARC as stated in Step •. The resulting GDG data set names would be SYS1.MQARC.CSQ1.SYSBMC.G0001V00 through SYS1.MQARC.CSQ1.SYSBMC.G0255V00, incrementing by one each time the solution is invoked. Note that you need to run QMARCDEF once for each queue manager that the System Queue Archival solution monitors, each time adjusting the GDG base data set name by changing the queue manager ID in the name.

- Third option: One GDG group per event queue per queue manager when the BMC Software MQSeries Event Listener is not running
 - A. Decide what name you want to give your GDG data set.
 - B. Add the queue manager and event type, separated by periods, to the data set name.
 - C. Specify **Unique(Y)** as an input parameter to EXEC QMQUNLDQ in rule MQARC001.
 - D. Using this option, if you chose SYS1.MQARC for the GDG base data set name, your queue manager ID is CSQ1 and your event type is QMGR. Then add .CSQ1.QMGR to the base name and specify this value for the data set specification in job QMARCDEF in BBSAMP. In Rule MQARC002, specify SYS1.MQARC (as stated in the following instructions). The resulting GDG data set names would be SYS1.MQARC.CSQ1.QMGR.G0001V00 through SYS1.MQARC.CSQ1.QMGR.G0255V00, incrementing by one each time the solution is invoked. Note that you need to run QMARCDEF once for each event type (QMGR, PERFM, CHANNEL) per queue manager that the System Queue Archival solution monitors, each time adjusting the GDG base data set name by changing the queue manager ID and event type in the name.

- Step 2** Invoke the Rule Processor application by typing **RULES** from any BBI terminal session screen.
- Step 3** Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
- Step 4** Select the AAORULBQ Rule Set.
- Step 5** Select the Rule MQARC002.
- Step 6** Edit the Rule using the Rule Processor application panels:
- On the Action Specification panel and in the **Set Variable** field, replace
- **GDGBASE****
- with the name of the GDG base specified in QMARCDEF (minus the ‘.CSQ1.SYSBMC’ and ‘.CSQ1.QMGR’ options previously referenced). Therefore, using the previous examples, you specify SYS1.MQARC for the variable in MQARC002.
- Step 7** By default, messages from the MQSeries offload utility are sent to the default SYSOUT class, or *. To use another class, edit the Rule MQARC003:
- On the Action Specification panel and in the **Set Variable** field, replace
- **OUTPUT****
- with the SYSOUT class to be used.
- Step 8** By default, the archival data set is allocated on the device SYSDA. To use another device, edit the Rule MQARC004:
- On the Action Specification panel and in the **Set Variable** field, replace
- **OUTUNT****
- with the device number to be used.
- Step 9** By default, the archival data set is allocated with the block size 0. To use another block size, edit the Rule MQARC005:
- On the Action Specification panel and in the **Set Variable** field, replace
- **OUTBLK****
- with the block size to be used.

Step 10 By default, the archival data set solution primary allocation is 190 tracks, and secondary allocation is 19 tracks. To change the number of tracks, edit the Rule MQARC006:

On the Action Specification panel and in the **Set Variable** field, replace:

```
**OUTTRK**
```

with the number of primary tracks to be allocated or (<primary>, <secondary>).

Step 11 After editing the Rule, load the GDG base name into its variable by issuing the following BBI command:

```
.E MQ
```

Step 12 Select the Rule MQARC001.

Step 13 Edit the Rule using the Rule Processor application panels.

13.A On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command .RESET MQ xx.

13.B Add the **UNIQUE(Y)** parameter to the EXEC specification on the Action Specification panel if using the second or third option described for QMARCDEF on page 6-11.

Step 14 Enable the MQARC001 and MQARC002 rules and all necessary MQARC003-MQARC006 Rules by performing *one* of the following actions:

- Enable the Rules on the Rule Processor Detail Control panel.
- Use the (E)nable line command on the Ruleset Overview panel.
- Issue the BBI commands

```
.T RULE,ENA,MQARC001
.T RULE,ENA,MQARC002
.T RULE,ENA,MQARC003
.T RULE,ENA,MQARC004
.T RULE,ENA,MQARC005
.T RULE,ENA,MQARC006
```

Note: The three event queues are, by default, set to be archived when 50 percent full. To use a value other than 50 percent, change the

value that the variable IMFQEPFL is compared to in the Rule's selection criteria. This value is changed on the Selection Variable Criteria panel for Rule MQARC001. There is a compare entry on the panel for each of the three event queues.

When the Rule is enabled and an event queue fills to a certain point, the Rule fires and the messages in the queue are moved to an OS/390 generation data set. The system queue is then enabled to continue receiving messages.

Stopping the System Queue Archival Solution

To inactivate the solution, disable the Rules by performing one of the following actions:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the **(D)**isable line command on the Ruleset Overview panel.
- Issue the BBI commands

```
.T RULE,DIS,MQARC001  
.T RULE,DIS,MQARC002  
.T RULE,DIS,MQARC003  
.T RULE,DIS,MQARC004  
.T RULE,DIS,MQARC005  
.T RULE,DIS,MQARC006
```

Using the Service Interval Performance Solution

Use the Service Interval Performance solution when you want to be informed automatically if a queue has not been receiving messages at its set rate. For example, a queue might be defined to receive (MQPUT) messages at a rate of once every five minutes. If a time interval of six minutes passes without the queue receiving any messages, a Q_SERVICE_INTERVAL_HIGH event occurs when the next MQPUT is issued for the queue and a Rule fires, generating an ALERT to inform you of the situation.

Once the solution is enabled, the ALERT informs you that the time interval has been exceeded for the queue. When the time interval and interval rate return to normal, the solution automatically deletes the ALERT.

Rule ID	Triggering Event
MQINT001	Fires when a Q_SERVICE_INTERVAL_HIGH event occurs.
MQINT002	Fires when a Q_SERVICE_INTERVAL_OK event occurs.

Security Considerations

To use the Service Interval Performance solution, MAINVIEW AutoOPERATOR needs the following access authority to MQSeries resources: Read access to any queues specified in the AAOMQLxx member of BBPARM.

Implementing the Service Interval Performance Solution

To invoke any part of the Service Interval Performance solution, first

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx.
2. Issue the .RESET MQ xx command where xx is the two-character suffix of the AAOMQLxx member.

Enabling the Service Interval Performance Rules

To invoke the Service Interval Performance solution, edit and enable the Rules MQINT001 and MQINT002 as follows:

- Step 1** Invoke the Rule Processor application by typing RULES from any BBI terminal session screen.
- Step 2** Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
- Step 3** Select the AAORULBQ Rule Set.
- Step 4** Edit MQINT001 using the Rules Processor application panels.

Press **Enter** to navigate through the panels and make the following changes:

- 4.A** On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command **.RESET MQ xx**.
- 4.B** Make sure that the MQSeries queues that you want to monitor have their QSVCINT and QSVCIEV attributes set accordingly. Refer to the *IBM MQSeries Command Reference* for more information.

- Step 5** Repeat Step 4 for MQINT002.
- Step 6** Enable the MQINT001 and MQINT002 Rules by performing *one* of the following steps:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E)nable line command on the Ruleset Overview panel.
- Issue the BBI commands:

```
.T RULE,ENA,MQINT001
```

```
.T RULE,ENA,MQINT002
```

Once each Rule is enabled, when a Q_SERVICE_INTERVAL_HIGH event occurs, the Rule fires and an ALERT is created. The ALERT states that the time interval has been exceeded for the queue.

When a Q_SERVICE_INTERVAL_OK event occurs (signaling that the interval rate has been satisfied), another Rule fires and deletes the ALERT.

Stopping the Service Interval Performance Solution

To inactivate the solution, disable Rules MQINT001 and MQINT002 by performing one of the following steps:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the (D)isable line command on the Ruleset Overview panel.
- Issue the BBI commands

```
.T RULE,DIS,MQINT001
.T RULE,DIS,MQINT002
```

Using the Queue Depth Management Solution

Use the Queue Depth Management solution when you want to be informed automatically of any user queue having a Q_DEPTH_HIGH event. A Q_DEPTH_HIGH event occurs when a queue has exceeded its depth threshold as defined by the QDEPTHHI attribute.

When a Q_DEPTH_HIGH EVENT occurs, a Rule fires, generating an ALERT to inform you that the queue has reached its depth threshold. The ALERT text will indicate which queue manager the queue in question is running under.

The Queue Depth Management solution also uses a second Rule, which fires when a Q_DEPTH_LOW condition occurs. This second Rule will delete the ALERT generated by the first Rule.

Rule ID	Triggering Event
MQQDP001	Fires when a Q_DEPTH_HIGH event occurs.
MQQDP002	Fires when a Q_DEPTH_LOW event occurs.

Security Considerations

To use the Queue Depth Management solution, MAINVIEW AutoOPERATOR needs the following access authority to MQSeries resources: Alter and Read access to queues specified in the AAOMQLxx member of BBPARM that participate in this monitoring.

Implementing the Queue Depth Management Solution

To invoke the Queue Depth Management solution, first

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx.
2. Issue the `.RESET MQ xx` command where `xx` is the two-character suffix of the AAOMQLxx member.

Enabling the Queue Depth Management Rules

To invoke the Queue Depth Management solution, edit and enable the Rules MQQDP001 and MQQDP002 as follows:

- Step 1** Invoke the Rule Processor application by typing **RULES** from any BBI terminal session screen.
- Step 2** Ensure that the AAORULBQ Rule Set is enabled. If not, enable the Rule Set by typing the (E)nable line command next to the Rule Set name.
- Step 3** Select the AAORULBQ Rule Set.
- Step 4** Edit MQQDP001 using the Rules Processor application panels.

Press **Enter** to navigate through the panels and make the following changes:
 - 4.A** On the Selection Criteria panel under Queue Identification, change XXXX to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command **.RESET MQ xx**.
 - 4.B** Make sure that the MQSeries queues that you want to monitor have their QDEPTHHI and QDEPTHLO attributes set accordingly. See the *IBM MQSeries Command Reference* for more details. For the solution to work properly, the percentage specified for QDEPTHLO should be one percent less than the percentage specified for QDEPTHHI.
- Step 5** Repeat Step 4 for MQQDP002.

Step 6 Enable the MQQDP001 and MQQDP002 Rules by performing *one* of the following:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the (E)nable line command on the Ruleset Overview panel.
- Issue the BBI commands:

```
.T RULE,ENA,MQQDP001  
.T RULE,ENA,MQQDP002
```

Once the Rule is enabled, when a Q_SERVICE_INTERVAL_HIGH event occurs, the Rule fires and an ALERT is created stating that the service interval has been exceeded for the queue.

When a Q_SERVICE_INTERVAL_OK event occurs (signaling that the interval rate has been satisfied), another Rule fires and deletes the ALERT.

Stopping the Queue Depth Management Solution

To inactivate the solution, disable Rules MQQDP001 and MQQDP002 by performing *one* of the following steps:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the (D)isable line command on the Ruleset Overview panel.
- Issue the BBI commands:

```
.T RULE,DIS,MQQDP001  
.T RULE,DIS,MQQDP002
```

Using the Channel Availability Solution

The Channel Availability solution is a collection of Rules that fires when CHANNEL_START and CHANNEL_STOP instrumentation events are generated for LU6.2 and TCP/IP channels.

When a CHANNEL_STOP event occurs, the Rules check if the event was due to one or more of the most common channel errors for MQSeries. In each case an ALERT is generated, indicating the type of stoppage, where it occurred and why. ALERT help panels accompany the ALERT text for all but the most obvious conditions. These panels describe the error in detail and prescribe steps to take to correct the channel outage.

When a CHANNEL_START event occurs, Rules fire that ensure ALERTs previously issued for the channel just started are deleted from the ALERT queue. This helps to keep availability information timely and accurate.

Users can choose those outage conditions they wish to monitor. The table below describes the Rules included in the solution.

Rule ID	Default State	Remote Command Issued?	Triggering Event
MQP2023D	Enabled	No	Connection ID or Remote Listener Unavailable - connection refused
MQTCK001	Disabled	No	Inspect TCP channel availability
MQP2023C	Enabled	No	Connection ID or Remote Listener Unavailable - connection timed out
MQP20832	Enabled	No	Error receiving data from connection ID - Network is down
MQP2083C	Enabled	No	Error receiving data from connection ID - Time out due to hardware failure
MQP20636	Enabled	No	Error sending data to connection ID - connection reset by peer
MQP54500	Disabled	Yes	Channel disconnect interval exceeded - MAINVIEW AutoOPERATOR restarting channel
MQPRMT01	Enabled	No	Remote channel stopped by user command
MQP00001	Enabled	No	Host channel stopped by user command

Note: If a Rule issues remote commands, you must check the name of the command queue for queue managers on non-OS/390 platforms. If the command queue name is not SYSTEM.COMMAND.QUEUE, you must make a copy of the Rule. The command specified on the Rule Processor Automation Action panel must be modified to include a CQ (command queue) keyword naming the command input queue.

Security Considerations

To use the Channel Availability solution, MAINVIEW AutoOPERATOR needs access authority to MQSeries resources as follows:

- control access to channels on which you want this solution to start
- authority to issue DISPLAY and START channel commands on distributed queue managers

Implementing the Channel Availability Solution

To invoke the Channel Availability solution, first

1. Add the names of the queue managers which this solution will manage to BBPARM member AAOMQLxx.
2. Issue the .RESET MQ xx command where xx is the two-character suffix of the AAOMQLxx member.

Note: The Channel Solution does not work for CICS channels.

Enabling the Channel Availability Rule

To invoke the Channel Availability solution, edit and enable the Rules in Rule Set AAORULBH by following these steps:

- Step 1** If you want to use Rule MQTCK001 to inspect TCP channel availability
- 1.A** Copy member AAOMQMxx from BBPARM to the first data set in the subsystem BBPARM concatenation.
 - 1.B** Rename it to have the same suffix as the AAOPRMxx member for this subsystem. (For example, if the AAOPRMxx for this subsystem has a suffix of 12, the name of the AAOMQMxx member should be AAOMQM12.)
 - 1.C** Edit the AAOMQMxx member. Add a line for the name of each remote queue manager that uses TCP to communicate with the local queue manager. Type the name in column 1 of each line.

- 1.D** Alter the attributes of the TCP sender channels you want to monitor by typing

SHORTRTY(1)
SHORTTMR(60)
LONGRTY(0)
LONGTMR(0)

Refer to the *IBM MQSeries Command Reference* for the exact command syntax.

- Step 2** Invoke the Rule Processor application by typing **RULES** from any BBI terminal session screen.
- Step 3** Ensure that the AAORULBH Rule Set is enabled. If not, enable the Rule Set by typing the **(E)nable** line command next to the Rule Set name.
- Step 4** Select the AAORULBH Rule Set.
- Step 5** Edit each Rule using the Rules Processor application panels.

Press **Enter** to navigate through the panels and make the following changes:

- 5.A** On the Selection Criteria panel under Queue Identification, change **XXXX** to the queue manager IDs of the queue managers that this Rule will manage. Remember that these queue manager names must be specified in BBPARM member AAOMQLxx and then reset with the BBI command **.RESET MQ xx**.

- 5.B** If you are using the Rule MQTCK001, go to the Action Specification panel. In the EXEC Name/Parms field replace

XXXX

with the four-character name of the local queue manager.

- Step 6** Enable the MQTCK001 Rule to monitor specific outage conditions, or any of the other Rules by performing *one* of the following actions:

- Enable the Rule on the Rule Processor Detail Control panel.
- Use the **(E)nable** line command on the Ruleset Overview panel.
- Issue the BBI command

.T RULE,ENA,ruleID

where *ruleID* is the Rule ID of the Rule you want to enable.

- Step 7** When you enable the Rule MQTCK001, the BBI-SS PAS must be restarted (either WARM or COLD), which allows the Rule to become automatically active at BBI-SS PAS initialization.

Stopping the Channel Availability Solution

To inactivate the solution, disable the Rules by performing *one* of the following actions:

- Disable the Rule on the Rule Processor Detail Control panel.
- Use the **(D)isable** line command on the Ruleset Overview panel.
- Issue the BBI command

.T RULE,DIS,*ruleID*

where *ruleID* is the Rule ID of the Rule you want to disable.

Using the Basic Intercommunication Solution

The Basic Intercommunication solution maximizes the availability of connections between OS/390 and non-OS/390 queue managers by using event-driven automation.

Note: The Basic Intercommunication EXECs use REXX socket calls; therefore, they will not work on systems, which do not have IBM TCP/IP active. Interlink TCPAccess does not support REXX socket calls.

Security Considerations

To use the Basic Intercommunication solution, MAINVIEW AutoOPERATOR needs access authority to MQSeries resources as follows:

- control access to any channels that you want this solution to start
- Read and Alter authority for transmit queues that this solution will monitor
- authority to issue commands on distributed queue managers

Before You Begin

For this solution to work, the PATROL for WebSphere MQ product must be installed on each non-OS/390 target platform to which you are connecting. For more information about this product, refer to the BMC Software documents *PATROL for WebSphere MQ Administration Guide* and *PATROL for WebSphere MQ Planning and Implementation Guide*.

The Basic Intercommunication solution is made up of MAINVIEW AutoOPERATOR Rules that are driven by MQSeries events. The Rules monitor MQSeries events that occur on remote systems (OS/390 or non-OS/390), then the Rules call EXECs that take actions on local and remote queue managers.

Users have the ability to change the mode in which this solution operates from Diagnose (Diag) mode to Repair mode by issuing a BBI Journal message, which causes a shared variable to be set that tells the solution what mode it is operating in. This operating mode specifies whether EXECs scheduled by the Basic Intercommunication solution should try to repair discovered errors or just diagnose and report the errors for manual resolution. If the EXECs cannot repair queue manager connection errors, an ALERT is issued requesting that an operator manually repair the connection.

The table below describes the Rules from the Rule Set AAORULBR that are included in the Basic Intercommunication solution.

Table 6-1 Rules from the Rule Set AAORULBR (Part 1 of 2)

Rule ID	Default State	EXEC Scheduled	Triggering Event
MQCOR000	Enabled	QMQCORCF	Time-initiated Rule used to define whether the Basic Intercommunication solution runs in Diag or Repair mode. This Rule executes at BBI-SS PAS start up. The default value is Diag mode. To change the mode to Repair <ul style="list-style-type: none"> • Copy the Rule into another Rule and change Diag to Repair on the Action Specification panel. Disable the MQCOR000 Rule. • Issue the following message from the BBI Journal to cause a Rule to fire and change the mode to Repair for the duration of the SS or until you change it back: *Rule Set Var Mqmode Repair
MQCORMOD	Disabled	QMQCORCF	Journal Rule that responds to the *Rule Set Var Mqmode Diag/Repair journal message and changes the mode in which the solution works.
MQCOR001	Enabled	QMQCORCF	Fires when a Channel_Stopped condition with a non-zero reason code is detected by MAINVIEW AutoOPERATOR. QMQCORCF is invoked and attempts to verify that the channel is using the MQSeries triggering process. If the EXEC is running in Repair mode, QMQCORCF starts the channel manually.
MQCOR006	Enabled	QMQCORCF	Fires when an Unknown_Object_Name event is detected by MAINVIEW AutoOPERATOR and the object queue manager is the same as the local queue manager. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR007	Enabled	QMQCORCF	Fires when a Put_Inhibited event is detected by MAINVIEW AutoOPERATOR for the System.Command.Input queue. QMQCORCF is invoked and attempts to correct the situation if the EXEC is running in Repair mode.
MQCOR008	Enabled	QMQCORCF	Fires when a Get_Inhibited event is detected by MAINVIEW AutoOPERATOR for the System.Command.Input queue. QMQCORCF is invoked and attempts to correct the situation if the EXEC is running in Repair mode.

Table 6-1 Rules from the Rule Set AAORULBR (Part 2 of 2)

Rule ID	Default State	EXEC Scheduled	Triggering Event
MQCOR009	Enabled	QMQCORCF	Fires when an Xmit_Q_Usage_Error event is detected by MAINVIEW AutoOPERATOR. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR010	Enabled	QMQCORCF	Fires when a Get_Inhibited event is detected by MAINVIEW AutoOPERATOR and the queue for which the event occurred is a transmit queue that follows the naming convention expected by Command MQ On Ramp. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR011	Enabled	QMQCORCF	Fires when a Put_Inhibited event is detected by MAINVIEW AutoOPERATOR and the queue for which the event occurred is a transmit queue that follows the naming convention expected by Command MQ On Ramp. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR012	Enabled	QMQCORCF	Fires when an Unknown_Remote_Q_Mgr event is detected by MAINVIEW AutoOPERATOR. The Rule notifies the operator of the condition through an ALERT.
MQCOR013	Enabled	QMQCORCF	Fires when an Xmit_Q_Type_Error event is detected by MAINVIEW AutoOPERATOR. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR014	Enabled	QMQCORCF	Fires when a Def_Xmit_Q_Usage_Error event is detected by MAINVIEW AutoOPERATOR. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR015	Enabled	QMQCORCF	Fires when an Unknown_Def_Xmit_Q event is detected by MAINVIEW AutoOPERATOR. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR016	Enabled	QMQCORCF	Fires when a Not_Authorized event is detected by MAINVIEW AutoOPERATOR. QMQCORCF notifies the operator of the condition through an ALERT.
MQCOR017	Enabled	QMQCORCF	Fires when a Put_Inhibited event is detected by MAINVIEW AutoOPERATOR and the queue for which the error occurred is a remote command queue. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.
MQCOR018	Enabled	QMQCORCF	Fires when a Q_Mgr_Not_Active event is detected by MAINVIEW AutoOPERATOR. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode or notifies the operator of the condition through an ALERT.
MQCOR019	Enabled	QMQCORCF	Fires when a Q_Full event is detected by MAINVIEW AutoOPERATOR and the queue for which the error occurred is a transmit queue. QMQCORCF is invoked and attempts to repair the condition if the EXEC is running in Repair mode.

Invoking the Basic Intercommunication Solution

To invoke the Basic Intercommunication solution, do the following:

1. Add the names of the OS/390 queue managers that this solution will manage to BBPARM member AAOMQLxx.
2. Issue the .RESET MQ xx command where xx is the two-character suffix of the AAOMQLxx member

Note: The Rule that is triggered by CHANNEL_STOPPED events does not work for CICS channels.

Enabling the Basic Intercommunication Solution

To enable the Basic Intercommunication solution, verify that the Rules in the Rule Set AAORULBR and the Rule Set itself are enabled. By default, each Rule in the Rule Set is already enabled except MQCORMOD.

- To verify the Rule Set is enabled, follow these steps:
 1. Invoke the Rule Processor application by typing RULES from any BBI terminal session screen.
 2. Ensure that the AAORULBR Rule Set is enabled. If not, enable the Rule Set by entering the (E)nable line command next to the Rule Set name.
- To verify the Rules are enabled, follow these steps:
 1. Select the AAORULBR Rule Set.
 2. Verify the Rules are enabled.
 3. Press **Enter** to navigate through the panels and make the following changes:

On the Selection Criteria panel under Manager(s), change XXXX to the queue manager IDs of the OS/390 queue managers that each Rule will manage.
- To enable MQCORMOD, follow these steps:
 1. Select the AAORULBR Rule Set.
 2. Edit the Rule using the Rules Processor application panels.

- To change the operating mode of the solution, issue *one* of the following messages:
 - Issue the `*Rule Set Var Mqmode Diag` journal message if you want the solution to try to diagnose the connection errors.
 - Issue the `*Rule Set Var Mqmode Repair` journal message if you want the solution to try to diagnose and repair the errors.

Stopping the Basic Intercommunication Solution

To stop the Basic Intercommunication solution, disable the Rules by performing *one* of the following actions:

- Disable the Rule Set in the Automation Control panel.
- Take one of the following actions to disable each of the Rules individually:
 - Disable the Rule on the Rule Processor Detail Control panel.
 - Enter the (D)isable line command on the Ruleset Overview panel.
 - Issue the BBI command

.T RULE,DIS,rule ID

where *ruleID* is the name of the Rule that you want to disable.

Chapter 7 Command MQ Automation Power Line (APL)

Command MQ Automation Power Line (APL) provides the capability to issue commands from an EXEC to manage non-OS/390 MQSeries queue managers and receive responses to those commands without using MQSeries Channels. APL can only be invoked from an EXEC.

The APL EXEC requires

- PATROL Node Manager running on a non-OS/390 node
- at least one BBI-SS PAS containing MAINVIEW AutoOPERATOR 6.2
- TCP/IP level 3.1 or higher running on OS/390

APL must have authority for MQSeries on the computer with which it is communicating. For information about PATROL Node Manager, refer to the *PATROL for WebSphere MQ Administration Guide*.

The following statement shows the format of the APL EXEC in REXX:

```
"IMFEXEC SELECT E(QMQPOWER CMD(Commands) NODE(IPname)),  
"PORT(port) RM(qmgr) WAIT(sec) DEBUG(debug) HELP(help)",  
"RESP(response)) WAIT(YES)"
```

Note: The APL EXEC uses REXX socket calls; therefore, it will not work on systems that do not have IBM TCP/IP active. Interlink TCPAccess does not support REXX socket calls.

Table 7-1 describes the parameters of the EXEC.

Table 7-1 APL parameters (Part 1 of 2)

Parameter	Required?	Function	Notes
CMD	YES	Valid PATROL Node Manager command	<p>This parameter has no default value. The PATROL Node Manager command can be up to 256 characters in length. You must replace all blanks with plus signs.</p> <p>If MQSeries commands use lowercase or mixed case MQSeries object name, the name should be enclosed in single quotation marks. To pass a single-quote enclosed character string, it must also be enclosed in single quotation marks. That is, the quotation marks must be typed twice, for example: 'CMD(DIS+CHL("CSQ1.TO.remote")+ALL)</p> <p>PATROL Node Manager accepts any valid MQSeries command and any of the following PATROL Node Manager "EXEC" commands: EXCMD_START_Q_MGR EXCMD_STOP_Q_MGR EXCMD_INQUIRE_Q_MGR_NAMES</p> <p>Multiple MQSeries or Node Manager "EXEC" commands can be passed in one request. The commands must be separated by two consecutive colons (::).</p>
NODE	YES	PATROL Node Manager network node name or IP address	This parameter has no default value.
PORT	NO	PATROL Node Manager port number on its node	<p>If omitted, this parameter uses the 5000 default value.</p> <p>Valid values are 0-9999.</p>
RM	NO	Non-MVS queue manager with which APL communicates	If omitted, this parameter uses the default value equal to Userid.
WAIT	NO	Number of seconds to wait for the PATROL Node Manager response. This time applies to every TCP/IP send or response communication.	If omitted, this parameter uses a 60-second default value.
DEBUG	NO	To write debug information to the journal (YES, NO)	If omitted, this parameter uses NO as the default.

Table 7-1 APL parameters (Part 2 of 2)

Parameter	Required?	Function	Notes
RESPonse	NO	Response format	Can have one of three values: NO: Responses for commands will not be saved. YES: (default) All responses will be saved in APLNxx variables, contiguously without separation lines between them. DELIM: Each response begins and ends with a separation line.
HELP	NO	To write APL's call format to the journal (YES, NO)	If omitted, this parameter uses NO as the default. If HELP(YES) is provided, all other parameters are ignored. Calling QMQPOWER without parameters is the same as calling it with HELP(YES).
For more information about MQSeries, refer to the <i>IBM MQSeries Application Programming Reference</i> .			

Variables Returned from APL

The IMFCC TSO variable describes the success or failure of the EXEC that calls APL. The IMFRC TSO variable contains information about the success of APL. LOCAL variables APLCC and APLRC are also returned (and they contain additional information). For more information, refer to Table 7-2.

Table 7-2 APL Variables IMFRC, APLCC, and APLRC

Message in APLLNxx	IMFRC	APLCC	APLRC
PL0000I SUCCESS or MQSeries response	0	0	0
PATROL Node Manager Return code	8	4	PATROL Node Manager reason code
SOCKET(SOCKET) rc=nn SOCKET(CONNECT) rc=nn SOCKET(WRITE) rc=nn SOCKET(SELECT) rc=nn	8	8	TCP/IP return code
Error in EXCMD command	8	12	1
Unable to initialize SOCKET	8	12	3
Connection reset by peer	8	12	4
TIMED OUT	8	12	5
Unrecognized response from MQ agent	8	12	6
CMD parameter is not provided	8	16	51
NODE parameter is not provided	8	16	52
Debug parameter can be only YES or NO	8	16	53
Required AO for MQSeries is not active	8	16	54
AO for MQSeries services required not available on this level. Must be 5.1 or higher	8	16	55
Error in parameter structure	8	16	56
Unknown parameter	8	16	57
Error in RESPONSE parameter	8	16	58

Note: If more than one command is passed, IMFRC, APLCC and APLRC correspond to the highest value of APLCC in the set of returns or the highest level of APLRC if APLCC=0.

For more information about TCP/IP return codes, refer to the *IBM TCP/IP for MVS Application Programming Interface Reference*, SC31-7187-03.

For more information about MQSeries return codes, refer to *MQSeries for MVS/ESA Message and Codes*, GC33-0819-03.

Additional response information from the queue manager is returned in the LOCAL variables APLNOL and APLL1 through APLL xx , where xx is the last line of returned information. APLNOL contains the number of returned lines (which is equal to the value of xx). These variables must be retrieved using the IMFEXEC VGET command before they can be used.

Variable Name	Definition
APLNOL	Contains the number of returned lines in variables APLL1 through APLL xx
APLL1	Contains the first line of response information from the queue manager
APLL xx	Contains the last line of response information from the queue manager

Note: If more than one command is passed:

- APLNOL contains the total number of the response lines from all responses together.
- APLL xx contains the response information from all commands. Each command response begins from a new line.

Figure 7-1 shows that the variable APLNOL has a value of 13, which equals 13 lines of returned information from the queue manager.

Figure 7-1 Example of Information Returned in Variables APLNOL and APLL1 through APLL13

```

APLNOL = 13
AMQ8408: Display queue manager details
DESCR( ) DEADQ(DEAD.LETTER.)
DEFXMITQ CHADEXIT( )
COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)QMNAME(EPESIN)
TRIGINT(999999999)MAXHANDS(256)
MAXUMSGS(10000)AUTHOREV(DISABLED)
INHIBTEV(ENABLED)LOCALEV(ENABLED)
REMOTEEV(ENABLED)PERFMEV(ENABLED)
STRSTPEV(ENABLED)CHAD(ENABLED)
CHADEV(DISABLED)MAXMSGL(4194304)
MAXPRTY(9)CCSID(437)
CMDLEVEL(500)DISTL(YES)
SYNCPT

```

Examples

The following examples show the majority of the functions of the APL EXEC in REXX.

Example 1 – Displaying Attributes of Queue Manager

Example

```
"IMFEXEC SELECT EXEC(QMQPOWER",  
"NODE(137.72.4.17) RM(EPESIN)",  
"CMD(DIS+QMGR+ALL) PORT(5006) DEBUG(YES)) WAIT(YES)"
```

This request displays all the attributes of queue manager EPESIN.

Note: This example uses an MQSeries command; when MQSeries commands are used, a + sign is used to separate the command and its parameters.

Example 2 – Requesting Names of Queue Managers

Example

```
"IMFEXEC SELECT EXEC(QMQPOWER",  
"NODE(137.72.4.17) DEBUG(YES)",  
"CMD(EXCMD_INQUIRE_Q_MGR_NAMES)) WAIT(YES)"
```

This command requests the names of the queue managers defined in the Windows NT system by the address of 137.72.4.17. The EXEC returns their names in one of the local variables APLLN1 - APLLNnn, depending on the number of queue managers.

Example 3 – Start Queue Manager

Example

```
"IMFEXEC SELECT EXEC(QMQPOWER",  
"NODE(137.72.4.17) RM(EPESIN)",  
"CMD(EXCMD_START_Q_MGR) PORT(5005)) WAIT(YES)"
```

This request starts queue manager EPESIN.

Example 4 – Stop Queue Manager

Example

```
"IMFEXEC SELECT EXEC(QMQPOWER",  
"CMD(EXCMD_STOP_Q_MGR) PORT(5006) NODE(137.72.4.17)",  
"RM(EPESIN)) WAIT(YES)"
```

This request stops queue manager EPESIN.

Example 5 – Passing Multiple Commands

Example

```
"IMFEXEC SELECT EXEC(QMQPOWER",  
"CMD(EXCMD_INQUIRE_Q_MGR_NAMES::DIS"||,  
"+QMGR+ALL::DIS+Q(BMC.LISTENER.SUB)+CURDEPTH)",  
"NODE(172.20.40.243) RM(EPESIN) PORT(5000)",  
"DEBUG(YES) WAIT(180)) WAIT(YES)"
```



```

CLUSTER( )
QUEUE( SYSTEM.ADMIN.COMMAND.QUEUE )
CRTIME( 16.01.00 )
ALTTIME( 16.01.00 )
PUT( ENABLED )
DEFPSIST( NO )
MAXMSGL( 9000 )
NOSHARE
HARDENBO
RETINTVL( 999999999 )
NOTRIGGER
TRIGDPH( 1 )
QDEPTHHI( 80 )
QDPMAXEV( ENABLED )
QDPLOEV( DISABLED )
QSVCIEV( NONE )
DEFTYPE( PREDEFINED )
SCOPE( QMGR )
IPPROCS( 1 )
CURDEPTH( 0 )
PL9001I CMD 2 24 DIS Q( SYSTEM.ADMIN.COMMAND.QUEUE ) ALL
PL9000I CMD 3 1 EXCMD_START_Q_MGR
SUCCESS
PL9001I CMD 3 1 EXCMD_START_Q_MGR
PL9000I CMD 4 18 DIS QMGR ALL
AMQ8408: Display Queue Manager details.
DESCR( )
DEFXMITQ( )
CLWLEXIT( )
REPOS( )
COMMANDQ( SYSTEM.ADMIN.COMMAND.QUEUE )
CRDATE( 1999-07-08 )
ALTDATA( 1999-11-11 )
QMID( EPESIN_1999-07-08_16.00.52 )
MAXHANDS( 256 )
AUTHOREV( ENABLED )
LOCALEV( ENABLED )
PERFMEV( ENABLED )
CHAD( DISABLED )
CLWLEN( 100 )
CCSID( 437 )
CMDLEVEL( 520 )
SYNCPT
DEADQ( DEAD.LETTER.QUEUE )
CHADEXIT( )
CLWLDATA( )
REPOSNL( )
QMNAME( EPESIN )
CRTIME( 16.00.52 )
ALTTIME( 13.26.35 )
TRIGINT( 999999999 )
MAXUMSGS( 10000 )
INHIBTEV( ENABLED )
REMOOTEV( ENABLED )
STRSTPEV( ENABLED )
CHADEV( ENABLED )
MAXMSGL( 4194304 )
MAXPRTY( 9 )
PLATFORM( WINDOWSNT )
DISTL( YES )
PL9001I CMD 4 18 DIS QMGR ALL
PL9000I CMD 5 2 DIS Q( BBOMVAO.YXP.NT ) CURDEPTH
AMQ8409: Display Queue details.
QUEUE( BBOMVAO.YXP.NT )
CURDEPTH( 17 )
PL9001I CMD 5 2 DIS Q( BBOMVAO.YXP.NT ) CURDEPTH

```

Chapter 8 Securing MAINVIEW AutoOPERATOR for MQSeries

The following resources for MQSeries for MVS/ESA can be secured:

- connections to MQSeries
- access to MQSeries objects (such as queues)
- use of MQSeries system commands

This security is implemented from within the MQSeries product. Therefore, when you want to use the MAINVIEW AutoOPERATOR for MQSeries component for automating MQSeries events, you must ensure MAINVIEW AutoOPERATOR for MQSeries is granted the proper authority to access queue managers, system queues, and user queues as required.

MAINVIEW AutoOPERATOR for MQSeries does allow you to specify who has access to updating and reading Rules and scheduling EXECs. If you need to secure who has access to updating and reading Rules and EXECs for MQSeries events, you must secure who has access to these resources. Refer to the BMC Software manual *Implementing Security for MAINVIEW Products* for more information about these security issues.

MAINVIEW AutoOPERATOR for MQSeries provides a new resource name:

prefix.ssid.AAO.target.APPL.MQSERIES

Use this resource name to secure who has access to the MAINVIEW AutoOPERATOR for MQSeries Workstation (described in Chapter 4, “Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics”).

For information about how to secure access to this resource, refer to the *Implementing Security for MAINVIEW Products* book and read about “Securing Resources for MAINVIEW AutoOPERATOR: Advanced Security” and “Securing MAINVIEW AutoOPERATOR Applications: Feature=APPL”.

Chapter 9 Using MAINVIEW AutoOPERATOR for MQSeries IMFEXEC Statements

This chapter contains the following discussions:

- description of various IMFEXEC MQI command statements that you can use in EXECs to interface with MQSeries objects
- description of IMFEXEC MQI variables
- description of the IMFEXEC CMD TYPE(MQS) statement that you can use to issue commands to the MQI command server
- description of the IMFEXEC COPY MQI and IMFEXEC DISPLAY MQI command statements that you can use to copy or display MQI variable values

For additional information, refer to the publication, *IBM MQSeries Application Programming Reference Guide*.

For information about other IMFEXEC command statements, refer to the *MAINVIEW AutoOPERATOR Advanced Automation Guide*.

MQI Commands

Using EXECs, you can use the MAINVIEW AutoOPERATOR for MQSeries EXEC interface to communicate with MQSeries objects. Table 9-1 briefly describes the IMFEXEC command statements and where you can find additional information.

Table 9-1 IMFEXEC Command Statements

Command	Function	Refer to
IMFEXEC MQI BACK	Back out changes since the last syncpoint	"MQI BACK" on page 9-13
IMFEXEC MQI CLOSE	Release access to an object	"MQI CLOSE" on page 9-15
IMFEXEC MQI CMIT	Commit all changes since last syncpoint	"MQI CMIT" on page 9-17
IMFEXEC MQI CONN	Connect to a queue manager	"MQI CONN" on page 9-19
IMFEXEC MQI DISC	Disconnect from a queue manager	"MQI DISC" on page 9-21
IMFEXEC MQI GET	Get a message from a local queue	"MQI GET" on page 9-23
IMFEXEC MQI OPEN	Establish access to an object	"MQI OPEN" on page 9-33
IMFEXEC MQI PUT	Put one or more messages into a queue	"MQI PUT" on page 9-37
IMFEXEC MQI PUT1	Put one message in a queue	"MQI PUT1" on page 9-47

In addition to the MQI commands, other function are available. They are COPY MQI and DISPLAY MQI. For detailed information about these commands, see "COPY MQI" on page 9-63 and "DISPLAY MQI" on page 9-66.

IMFEXEC MQI Variables

The following list describes general properties for the IMFEXEC MQI variables:

- Variable names consist of a
 - prefix name (for example, IMFMQI)
 - structure name (for example, MD, DLH)
 - field name (for example, **MsgId** or **CorrelId** in the message descriptor)
- Each node of the variable name is delimited by an underscore (_)

- Field-name variables whose corresponding possible values are constants, defined for them by MQSeries, are populated with the constant name rather than the actual value. For example, in the case of a Dead Letter message, the variable `IMFMQI_MD_FORMAT`, which contains the format field from the message descriptor, is set to `MQFMT_DEAD_LETTER_HEADER`. Likewise, the variable `IMFMQI_MD_VERSION`, which contains the version of the message descriptor, is set to `MQMD_VERSION_1` instead of `x'1'`.

The following rules apply for variables set by MAINVIEW AutoOPERATOR for MQSeries:

- If the value for a character field does not have a defined constant value, the actual value is used.
- If the value for a numeric field that has a defined range does not have a defined constant, the actual numeric value converted to zoned decimal is used.
- Variables for fields that are defined as hexadecimal fields (`MQBYTExx` in MQSeries) are not converted and are set 'as is.'
- Variables that are created for flag fields in a MQSeries structure may contain multiple constant values, delimited by blanks. For example, the variable for the message descriptor Report field, `IMFMQI_MD_REPORT`, can have several values. The resulting value might look like this: `'MQRO_COA MQRO_COD'`. Any other valid combination could be seen as well.
- Variables that are created for MQSeries fields where the value has more than one defined constant will contain both constants. For example, the variable `IMFMQI_MD_PUTAPPLTYPE`, which contains the PutApplType value from the message descriptor, would contain the value, `'MQAT_OS390 MQAT_MVS'`, if the message originated on an OS/390 system. This result is because both constants contain the value '2'.

What the MAINVIEW AutoOPERATOR MQI EXEC Interface Is

The MAINVIEW AutoOPERATOR MQI EXEC interface consists of the following features:

- commands through which applications can access the queue manager and its facilities
- commands and variable structures that applications use to pass data to and get data from the queue manager
- elementary data types for passing data to, and getting data from, the queue manager

The MQI commands in the MAINVIEW AutoOPERATOR MQI EXEC interface correspond to MQSeries MQI functions.

How Applications Communicate with MQSeries Objects

For an EXEC to communicate with a queue manager, the application must have a unique identifier by which it knows that queue manager. This identifier is called a *connection handle*. The connection handle is returned by the IMFEXEC MQI CONN statement when an application connects to the queue manager. This connection handle is used as an input, explicitly or implicitly, when an application executes subsequent MQI statements.

How Applications Interact with MQSeries Objects

For an application to work with a MQSeries object (for example: a queue, a name list, a queue manager, or a process definition), both the application and the queue manager must have a unique identifier by which it knows that object. This identifier is called an *object handle*. The handle is returned by the IMFEXEC MQI OPEN statement when an application opens the object to work with it. Applications pass the object handle as input, explicitly or implicitly, when they use subsequent IMFEXEC MQI PUT, GET, or CLOSE statements.

How Completion and Reason Codes Are Returned

A completion code and reason code are returned as output by each statement. The completion code is stored in the IMFCC and IMFMQCC variables. The variables usually contain a 0 or 2 in IMFMQCC and a 0 or 8 in IMFCC, showing success and failure, respectively. Some IMFEXEC MQI statements can return an intermediate state, a 1 in IMFMQCC and a 4 in IMFCC, indicating partial success.

The reason code is stored in the IMFRC and IMFMQRC variables. The contents of both variables are the same. The reason code shows the reason for the failure, or partial success. There are many reason codes covering circumstances such as a full queue or GET operations not allowed for a queue. Applications can analyze the reason code and based on the returned value, take additional actions.

The variable IMFMQI_REASON contains the constant (character string) reason code that corresponds with the numeric reason code returned by MQSeries when an error is encountered. For example: when MQSeries returns the reason code 2195, the variable IMFMQI_REASON contains 'MQRC_UNEXPECTED_ERROR'. The numeric reason code is displayed on the EM9201I message, which is sent to the BBI journal when there is an error returned by MQSeries.

The completion codes for each statement with the description of the statement are listed in Table 9-2.

Table 9-2 MQI EXEC Completion Codes (Part 1 of 2)

IMFMQCC	IMFCC	Reason	System Action	User Action
0	0	Successful completion	None	None
2	8	MQSeries returned error	Command fails	You have 2 options: <ul style="list-style-type: none"> • Obtain the numeric reason code from variable IMFMQRC and refer to the <i>IBM MQSeries Application Programming Reference Guide</i> for reason information. • Obtain the character constant reason code from variable IMFMQI_REASON and take appropriate action. The value of this variable corresponds to the numeric value of IMFMQRC. For example, if IMFMQRC = 2033, then IMFMQI_REASON = 'MQRC_NO_MSG_AVAILABLE'.

Table 9-2 MQI EXEC Completion Codes (Part 2 of 2)

IMFMQCC	IMFCC	Reason	System Action	User Action
	12	Buffer too small	Message is not PUT to the queue and command fails.	Specify a buffer large enough to hold the entire message and all included structures. See the topic "Creating Messages from Variables" for more information.
4	16	Syntax error	Command fails	Refer to EM0022E message in BBI journal for error description.
	20	Variable Processing routine not found	Command fails; for COPY MQI and DISPLAY MQI commands only	Look for errors during BBI PAS startup, related to program QAVARS, correct if possible and restart the BBI PAS. Otherwise contact BMC Software customer support.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. For the reason code descriptions, refer to the "Message Queuing Constants" chapter of the *IBM MQSeries Application Programming Reference Guide*.

Creating Messages from Variables

You can use MAINVIEW AutoOPERATOR for MQSeries variables and IMFEXEC MQI statements to modify existing MQSeries messages. You can also use MAINVIEW AutoOPERATOR for MQSeries to create new MQSeries messages. This section describes how you can accomplish these tasks.

The IBM MQSeries product offers a programming interface for many languages such as assembler, COBOL and C. BMC Software has created the IMFEXEC MQI interface for MAINVIEW AutoOPERATOR EXECs to provide access to these facilities using the REXX and CLIST programming languages. The structures associated with the IBM MQSeries API (application programming interface) are available to you through MAINVIEW AutoOPERATOR for MQSeries variables in the format

IMFMQI_structure_fieldname

where *structure* is the structure identifier and *fieldname* is the name of the field in the structure. See the section "IMFEXEC MQI Variables" on page 9-2 for more information about the format and content of these variables.

MAINVIEW AutoOPERATOR can update messages retrieved from a queue, based upon values of variables that you set in your EXEC and then place the updated message on the same or a different queue when the IMFEXEC MQI PUT or PUT1 statement is executed. Some variables correspond to each field of each structure in MQSeries (CIH, DLH, IIH, MD, MDE, RFH, RFH2, RMH, TM, TMC2, WIH, XQH, CFH, OD).

Additionally, other variables correspond to each data item name for an instrumentation event. Details about the most commonly used variables for the IMFEXEC MQI statements are covered in this chapter; all other variables are listed in Appendix D, “EXECs Variables” in this book.

When creating new messages, MAINVIEW AutoOPERATOR builds the Message Descriptor, the Object Descriptor, and the message buffer from variables that you create, or from default values where possible (there are no default values for the message buffer). Note that when creating new messages, the author of the EXEC is responsible for entering all necessary variables for structures except for the Message Descriptor and Object Descriptor (MD and OD).

Creating a Simple Message

MAINVIEW AutoOPERATOR for MQSeries creates variables from messages it reads from queues when executing the IMFEXEC MQI GET statement. Likewise, new messages created by an EXEC, for the purpose of being PUT or PUT1 to a queue, are built from the contents of variables you create in the EXEC. An example of a message with a Message Descriptor and no additional structures is pictured in Figure 9-1.

Figure 9-1 Simple Message

Message Descriptor

```
IMFMQI_MD_FORMAT
='MQFMT_STRING'
```

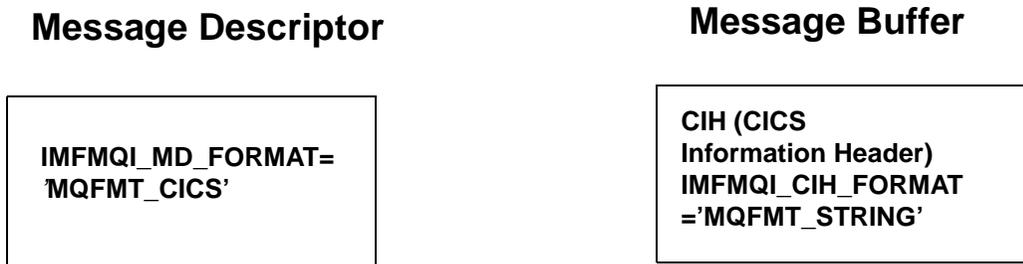
Message Buffer

```
Application
Data
```

Creating a Complex Message

When creating new messages, you can add MQSeries structures to the messages by changing the value of the message descriptor format variable, `IMFMQI_MD_FORMAT`, and creating the necessary structure variables. A message with a message descriptor and an added structure is shown in Figure 9-2.

Figure 9-2 Complex message



Updating an Existing Message

You can update existing messages (ones which have been retrieved from a queue by the `IMFEXEC MQI GET` statement) by updating the variables created during the `GET`, and `PUT`ting the message to the same or a different queue. Additionally, a new structure can be added to an existing message. If you decide to add a structure to a message, the message buffer provided must be large enough to hold the new structure and a position in the message buffer must be available to write the data. When you add a structure (`DLH`, `CIH`, `WIH`, etc.), the following steps must be taken:

- Step 1** Update the variable `IMFMQI_MD_FORMAT` to specify the format of the structure you are adding, for example:

```
MQFMT_DEAD_LETTER_HEADER
or
MQFMT_WORK_INFO_HEADER
```

- Step 2** Create the appropriate structure variables.

You must create all the variables in the structure because `MAINVIEW AutoOPERATOR` will only update the message buffer for each field's position if the corresponding variable is set.

- Step 3** Create a new message buffer that is large enough to hold the new structure you are adding and the remaining current message buffer.

For example, suppose you are adding the WIH (Work Information Header) structure that is 120 bytes and the current message buffer is called IMFMQI_BUFFER. You could create a variable that consists of the WIH structure ID ('WIH ') followed by 116 blanks (call it WIH, for example). You would issue the following REXX instruction in your EXEC to create the new message buffer:

```
My_New_Buffer = WIH | | IMFMQI_BUFFER
```

- Step 4** Issue the IMFEXEC MQI PUT or PUT1 statement (assuming that you have connected to the queue manager and opened the destination queue).

Note: If the buffer specified is not large enough, an error code of 12 will be returned to the EXEC in the variable IMFCC and the PUT or PUT1 statement will not be completed.

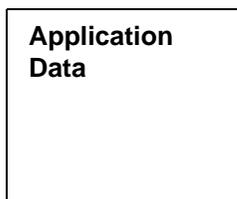
Adding a Structure to an Existing Message

Adding a new structure to an existing message is more difficult than updating an existing message since the design of MAINVIEW AutoOPERATOR for MQSeries, in this area, is more suited for updating existing structures. With existing messages, MAINVIEW AutoOPERATOR is dealing with a message buffer that already contains the structures in place. However, to add new structures, a new message buffer must be created.

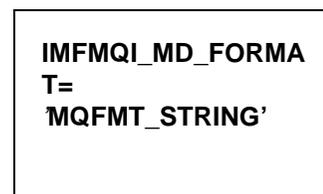
In this example, you want to GET a message from a queue, add a Dead Letter Header to it and PUT it onto the dead letter queue. After issuing the IMFEXEC MQI GET statement, the message buffer (IMFMQI_BUFFER) by default, contains the application data, as shown in Figure 9-3.

Figure 9-3 Existing Message

Message Buffer



Message Descriptor



Then, your EXEC sets Message Descriptor (MD), Dead Letter Header (DLH) and buffer variables as follows:

```

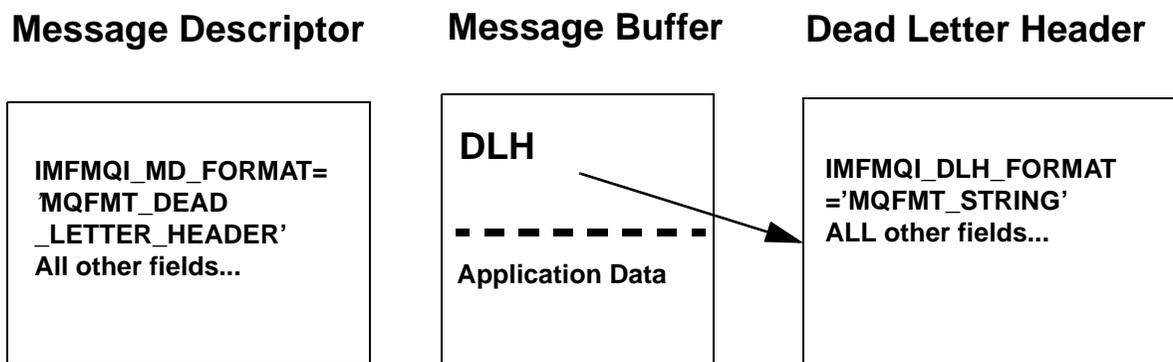
IMFMQI_MD_FORMAT = 'MQFMT_DEAD_LETTER_HEADER'
IMFMQI_DLH_STRUCID = 'MQDLH_STRUC_ID'
IMFMQI_DLH_VERSION = 'MQDLH_VERSION_1'
IMFMQI_DLH_FORMAT = 'MQFMT_STRING'
IMFMQI_DLH_REASON = '915'
IMFMQI_DLH_DESTQNAME = 'MY.QUEUE1'
IMFMQI_DLH_DESTQMGRNAME = 'MYREMOTE'
IMFMQI_DLH_ENCODING = 'MQENC_NATIVE'
IMFMQI_DLH_CODEDCHARSETID = '500'
IMFMQI_DLH_PUTAPPLTYPE = 'MQAT_OS390'
IMFMQI_DLH_PUTAPPLNAME = 'MYAPPL'
IMFMQI_MD_PUTDATE = date('s')
t = substr(translate(time('L'),' ',':.'),1,11)
IMFMQI_MD_PUTTIME =
substr(t,1,2)||substr(t,4,2)||substr(t,7,2)||substr(t,10,2)
My_New_Buffer = DLH||IMFMQI_BUFFER./* DLH = 'DLH...blanks...'* /

```

To use the new message buffer, issue IMFEXEC MQI PUT or PUT1 using the parameters, BUFFER(My_New_Buffer) and BUFFLEN(length(My_New_Buffer)).

During processing of the IMFEXEC MQI PUT or PUT1 statement, the Message Descriptor, Message Buffer and Dead Letter Header will be updated as shown in Figure 9-4.

Figure 9-4 Updating Message Buffer and Dead Letter Header



Note: When processing messages, be aware that

- To add structures to your message, the message buffer must be large enough to hold the structure and the contents of the buffer. If it is not large enough, an error code 12 is returned to the EXEC in the variable IMFCC.
- You can chain multiple structures together by updating the Format field of the previous structure, as long as the message buffer is the correct size and the space that will contain the structure is empty.
- You cannot chain multiple structures together when a CFH structure is used.
- You cannot create instrumentation events with an EXEC.

Adding an IMS Bridge Message

In this example, an IMS bridge message is built. The necessary object descriptor variables and all the IIH structure variables are set as follows:

```

IMFMQI_OD_OBJECTNAME = QUEUE
IMFMQI_OO_OPTIONS = 'M000_OUTPUT'
IMFMQI_MD_REPLYTOQ = "BBOMVAO.YXP.QUEUE1"
IMFMQI_MD_REPLYTOQMGR = lm
IMFMQI_MD_PERSISTENCE = "MQPER_PERSISTENT"
IMFMQI_MD_FORMAT = "MQFMT_IMS"
IMFMQI_PMO_TIMEOUT = -1
/* Variable Name value namevalue length*/
IMFMQI_IIH_STRUCID= "MQIIH_STRUC_ID"/* "IIH" 4*/
IMFMQI_IIH_VERSION= "MQIIH_VERSION_1"/* "00000001"X 4*/
IMFMQI_IIH_STRUCLength= "MQIIH_LENGTH_1"/* "00000054"X 4*/
IMFMQI_IIH_ENCODING= "MQENC_NATIVE"/* 785 4*/
IMFMQI_IIH_CODEDCHARSETID = "MQCCSI_Q_MGR"/* "00000000"X 4*/
IMFMQI_IIH_FORMAT= "MQFMT_IMS_VAR_STRING"/* "MQIMSVS " 8*/
IMFMQI_IIH_FLAGS= "MQIIH_NONE"/* "00000000"X 4*/
IMFMQI_IIH_LTERMOVERRIDE = " "/* 8*/
IMFMQI_IIH_MFSMAPNAME= " "/* 8*/
IMFMQI_IIH_REPLYTOFORMAT = "MQFMT_NONE"/* " " 8*/
IMFMQI_IIH_AUTHENTICATOR = "MQIAUT_NONE"/* " " 8*/
IMFMQI_IIH_TRANINSTANCEID = copies("00"X,16)/* 16*/
IMFMQI_IIH_TRANSTATE= "MQITS_NOT_IN_CONVERSATION" /* " " 1*/

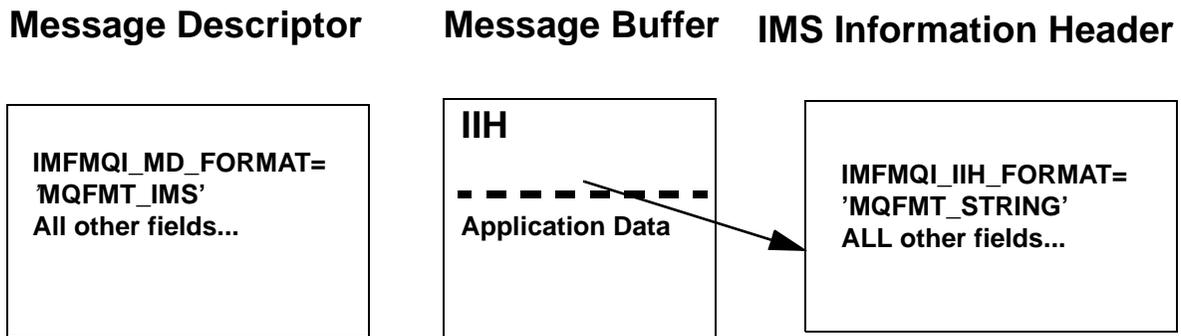
```

```

IMFMQI_IIH_COMMITMODE= "MQICM_SEND_THEN_COMMIT" /* "1" 1*/
IMFMQI_IIH_SECURITYSCOPE = "MQISS_CHECK" /* "C" 1*/
IMFMQI_IIH_RESERVED= " " /* " " 1*/
LL          = "0050"X
ZZ          = "0000"X
TRANCODE    = LEFT("GBGTRNM1 .",76)
PUTDATA = left(' ',84) || ll || zz ||TRANCODE
bufl = length(putdata)
"IMFEXEC MQI PUT1 BUFFER(PUTDATA) " || ,
"BUFFLEN("BUFL") POPTS('MQPMO_NO_SYNCPOINT')"
```

During the IMFEXEC MQI PUT or PUT1 statement processing, the Message Descriptor, Message Buffer and IMS Information Header are created as in Figure 9-5.

Figure 9-5 Creating a Message Descriptor, Message Buffer and an IMS Information Header



MQI BACK

This statement invokes the MQSeries BACK command which backs out MQ message changes since last syncpoint for all queries opened with SYNCPoint.

Command	Parameters
IMFEXEC MQI BACK	HCONN(hconn)

The following table describes the IMFEXEC MQI BACK parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> specifies a variable created in an earlier CONN command is not required <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.</p>

Example

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"           /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = "TEST.QUEUE1"   /* Set queue name */
"IMFEXEC MQI OPEN OOPTS('MQOO_OUTPUT') /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)"       /* Put message on queue*/
/* Wait for another EXEC to post this EXEC and if it fails, backout changes*/
"IMFEXEC WAIT 30 NAME(MQPUTECB1)"       /* Wait on Post*/
IF IMFPOST /= 'PUTECB1' THEN
    "IMFEXEC MQI BACK"                   /* Back out the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"    /* Close the queue */
"IMFEXEC MQI DISC"                       /* Disconnect from queue manager */
EXIT

```

This command backs out all the changes that occurred since the last syncpoint (previous **BACK** or **COMMIT**). The following list describes the variable values that are used or set by this statement:

- Variable **IMFHCONN** contains the connection handle name.
- Variable **IMFMQCC** and **IMFCC** contain the completion code.
- Variable **IMFMQRC** and **IMFRC** contain the reason code.
- Variable **IMFHOBJ** contains the object handle for the queue.
- Variable **IMFMQL_REASON** contains the constant (character) reason code.

MQI CLOSE

This statement closes a previously opened queue.

Command	Parameters
IMFEXEC MQI CLOSE	HCONN(hconn) HOBJ(hobj) COPTS(options)

The following table describes the IMFEXEC MQI CLOSE parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> Specifies a variable created in an earlier CONN command Is not required <p>If omitted, the connection handle value is the default variable, IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.</p>
HOBJ	Provides the object handle that represents the object being closed.	<p>This parameter</p> <ul style="list-style-type: none"> Specifies a variable created in an earlier OPEN command Is not required <p>If omitted:</p> <ul style="list-style-type: none"> The object handle value is taken from the default variable IMFHOBJ. The object handle is returned by a previous IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable.
COPTS	Specifies the options that control the action of CLOSE.	<p>This parameter is not required.</p> <p>If omitted, the close options value is taken from the default variable IMFMQI_CO_OPTIONS.</p> <p>If IMFMQI_CO_OPTIONS is not set, the default option of MQCO_NONE is used.</p> <p>Valid close options:</p> <p>MQCO_NONE MQCO_DELETE MQCO_DELETE_PURGE</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"/ * Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1/* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"/ * Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT'/* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
IMFMQI_BUFFERLEN = LENGTH(PUTDATA) /* Specify buffer length*/
IMFMQI_BUFFER = PUTDATA/* Place data in buffer*/
"IMFEXEC MQI PUT" /* Put message on queue*/
"IMFEXEC MQI COMMIT" /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"/ * Close the queue*/
"IMFEXEC MQI DISC" /* Disconnect from queue manager */
EXIT
```

This example highlights closing an object defined by IMFHOBJ for the queue manager whose connection handle is stored in variable IMFHCONN. The following list describes the variable values that are used or set by this command statement:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.

MQI CMIT

This statement invokes the MQSeries CMIT command which commits all new messages and changes to queues opened with SYNCPoint.

Command	Parameters
IMFEXEC MQI CMIT	HCONN(hconn)

The following table describes the IMFEXEC MQI CMIT parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier CONN command • Is not required <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"          /* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1    /* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"  /* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT' /* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
IMFMQI_BUFFER = PUTDATA               /* Place data in buffer*/
"IMFEXEC MQI PUT BUFFER(PUTDATA)"      /* Put message on queue*/
/* Wait for another EXEC to post this EXEC and if it fails, backout out
changes. */
"IMFEXEC WAIT 30 NAME(MQPUTECB1)"      /* Wait on Post*/
IF IMFPOST /= 'PUTECB1' THEN
  "IMFEXEC MQI BACK"                  /* Back out the work */
ELSE
  "IMFEXEC MQI COMMIT"                /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"   /* Close the queue*/
"IMFEXEC MQI DISC"                    /* Disconnect from queue manager */
EXIT
```

This example highlights committing all changes that have occurred since the last syncpoint. The following list describes the variable values that are used or set by this command statement:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.
- Variable IMFHOBJ contains the object handle for the queue.

MQI CONN

With the MQI CONN statement, an application can connect to a queue manager. It returns a queue manager connection handle to the EXEC, which is used by the application for all other IMFEXEC MQI statements.

Command	Parameters
IMFEXEC MQI CONN	NAME(qmgr) HCONN(hconn)

The following table describes the IMFEXEC MQI CONN parameters.

Parameter	Function	Notes
NAME	The name of a connectable queue manager.	
HCONN	Provides a variable to contain the connection handle.	This parameter <ul style="list-style-type: none"> • Specifies a variable where the connection handle will be placed • Is not required If omitted, the connection handle value is the default variable IMFHCONN.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"/* Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1/* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"/* Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT'/* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)" /* Put message on queue*/
"IMFEXEC MQI COMMIT" /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"/* Close the queue */
"IMFEXEC MQI DISC" /* Disconnect from queue manager */
EXIT
```

This example highlights connecting the EXEC to the queue manager CSQ1. The following list describes the variable values that are used or set by this command:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI_REASON contains the constant (character) reason code.

MQI DISC

With the MQI DISC statement, an application can disconnect from a queue manager.

Command	Parameters
IMFEXEC MQI DISC	HCONN(hconn)

The following table describes the IMFEXEC MQI DISC parameters.

Parameter	Function	Notes
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier CONN command • Is not required <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable.</p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same.

IMFMQI_REASON contains the constant (character) reason code. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"/ * Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1/* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"/ * Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT'/* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)" /* Put message on queue */
"IMFEXEC MSG 'MQEXAMPL - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'"
"IMFEXEC MQI COMMIT" /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"/ * Close the queue */
"IMFEXEC MQI DISC" /* Disconnect from queue manager */
EXIT
```

This example highlights disconnecting from the queue manager whose connection handle is stored in variable IMFHCONN. The following list describes the variable values that are used or set by this command:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI_REASON contains the constant (character) reason code.

MQI GET

This statement is used to GET a message from a previously opened queue.

Command	Parameters
IMFEXEC MQI GET	HCONN(hconn) HOBJ(hobj) GOPTS(options) BUFFLEN(bufferlength) BUFFER(buffer) DATALEN(datalength)

The following table describes the IMFEXEC MQI GET parameters.

Parameter	Function	Notes
<p>Note: The following list of constants may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the IBM MQSeries Application Programming Reference Guide.</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier CONN command • Is not required <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>
HOBJ	Provides the object handle that represents the queue from which the message is returned.	<p>This parameter:</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier OPEN command • Is not required <p>If omitted:</p> <ul style="list-style-type: none"> • The object handle value is taken from the default variable IMFHOBJ. • The object handle is returned by a previous IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable. <p>Either this parameter, specifying an existing variable that contains the object handle, or a previously created IMFHOBJ variable containing the object handle is required.</p>

Parameter	Function	Notes
GOPTS	Specifies the options that control the action of GET.	<p>This parameter is not required. If omitted, the GET options value is taken from the default variable <code>IMFMQI_GMO_OPTIONS</code>.</p> <p>The variable <code>IMFMQI_GMO_OPTIONS</code> is not required. If GET options are not specified, MQSeries will use the default, <code>MQGMO_NO_WAIT</code>.</p> <p>More than one of the following options can be specified in any valid combination.</p> <p>Valid GET options:</p> <ul style="list-style-type: none"> <code>MQGMO_WAIT</code> <code>MQGMO_NO_WAIT</code> <code>MQGMO_SYNCPOINT</code> <code>MQGMO_SYNCPOINT_IF_PERSISTENT</code> <code>MQGMO_NO_SYNCPOINT</code> <code>MQGMO_MARK_SKIP_BACKOUT</code> <code>MQGMO_BROWSE_FIRST</code> <code>MQGMO_BROWSE_NEXT</code> <code>MQGMO_MSG_UNDER_CURSOR</code> <code>MQGMO_ACCEPT_TRUNCATED_MSG</code> <code>MQGMO_SET_SIGNAL</code> <code>MQGMO_FAIL_IF QUIESCING</code> <code>MQGMO_CONVERT</code> <code>MQGMO_NONE</code> <p><i>See note at the beginning of this table.</i></p>
	Specifies the options that control the action of GET.	<p>In addition to the above options, another MAINVIEW AutoOPERATOR-provided option is available to the user. This option is <code>REMOVE_DLH</code>.</p> <p>Using the <code>REMOVE_DLH</code> option, you can retrieve messages from a Dead Letter queue, process them and forward them to an application queue without the Dead Letter Header. The <code>REMOVE_DLH</code> option is added to the <code>GOPTS()</code> keyword parameter of the <code>IMFEXEC MQI GET</code> statement. This MAINVIEW AutoOPERATOR option causes the <code>IMFMQI_MD_ENCODING</code>, <code>IMFMQI_MD_CODEDCHARSETID</code> and <code>IMFMQI_MD_FORMAT</code> variables to be set from the Dead Letter Header instead of the message descriptor. The returned buffer is without the DLH. This action occurs before control is returned to the EXEC following the <code>IMFEXEC MQI GET</code> command. The message can optionally be placed directly onto another queue.</p>
BUFFER	Specifies the variable that contains the message buffer data following a successful GET.	This parameter is not required. If omitted, the variable <code>IMFMQI_BUFFER</code> will contain the Message Buffer data upon successful completion of the GET.

Parameter	Function	Notes
BUFFLEN	Specifies the length of the data used to create the variable containing the message buffer data.	<p>This parameter</p> <ul style="list-style-type: none"> • Is not required • Can specify a numeric value or the name of a variable that contains a numeric value • Can specify a value in the range of 0 to 32767, either specifically, or using a variable <p>If this parameter is omitted, the variable IMFMQI_BUFFLEN is used to obtain the BUFFER length. If the variable is not set, the message length is used. The recommendation is that you do not set the variable so that the message length is used.</p> <p>Note: The BUFFER variable will be left justified and blank-padded to the right up to the value of the BUFFLEN variable. If you want to change the BUFFER variable size to be exactly the message size, use the REXX SUBSTR or LEFT built-in function. Examples follow:</p> <pre>IMFMQI_BUFFER = substr(IMFMQI_BUFFER,1,IMFMQI_DATALEN) or IMFMQI_BUFFER = LEFT(IMFMQI_BUFFER,IMFMQI_DATALEN)</pre>
DATALEN	Specifies the variable to be set to the message length following a successful GET.	<p>This parameter is not required.</p> <p>If omitted, the variable IMFMQI_DATALEN is set with the length of the message data returned from the GET. If the value is greater than the buffer length value, only buffer length bytes are returned in the buffer variable. If the value is zero, it means that the message contains no application data.</p>

The following table describes the most common input and output variables for the IMFEXEC MQI GET command. These variables, whether used as input or output, are always accessible after a successful GET. For structures other than the MD (message descriptor), refer to Appendix D, “EXECs Variables”. Variables created from structures begin with “IMFMQI_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore.

For definitions of field names and constants used in this table that are derived from MQSeries structure field names, see the *IBM MQSeries Application Programming Reference Guide*.

Variable Name	Source	Notes
<p>Note: The following list of constants may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>		
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures contained in the message. For example, in the case of a Dead Letter message, the contents would contain at least: 'MD DLH'
IMFMQI_OFFSETS	Output	Contains a blank-delimited list of the offsets of the structures contained in the message (as well as the offset of the application data) from the beginning of the message buffer. Using this variable, you can access application data beyond the structures in the message buffer. The number of entries in the list is always one more than the number of structures participating in the operation (IMFMQI_STRUCTURES) because an entry for the offset of the application data is included in the list. A 'NONE' is always used for a structure that does not exist in the buffer, such as 'MD'. For example, if the message was a dead letter message and contained some application data, the variable might look like this: 'NONE 0 172' In the above example, NONE is specified for the MD, 0 for the offset of the Dead Letter Header and 172 is the offset of the application data.
IMFMQI_MD_STRUCID	Input	(optional) The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_MD_VERSION	Input	(optional) The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used unmodified by the EXEC Manager.

Variable Name	Source	Notes
IMFMQI_MD_REPORT	Output	<p>Contains one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_MSGTYPE	Output	<p>Contains one of the following:</p> <p>MQMT_DATAGRAM MQMT_REQUEST MQMT_REPLY MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_EXPIRY	Output	<p>Contains a decimal number, in the range 1 to 999999999, or MQEI_UNLIMITED.</p>
IMFMQI_MD_FEEDBACK	Output	<p>Contains one of the following values for report messages:</p> <p>MQFB_NONE MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT</p> <p>In addition, this field can contain any reason code from the following sources:</p> <ul style="list-style-type: none"> • IMS-bridge feedback codes • CICS-bridge feedback codes • MQSeries reason codes <p><i>See note at the beginning of this table.</i></p>

Variable Name	Source	Notes
IMFMQI_MD_ENCODING	Output	Contains MQENC_NATIVE Or any decimal number up to 999999999.
IMFMQI_MD_CODEDCHARSETID	Input/ Output	Contains MQCCSI_Q_MGR MQCCSI_EMBEDDED Or any decimal number up to 999999999. See <i>IBM MQSeries Application Programming Reference Guide</i> for details on how to use the CodedCharSetId field as input.
IMFMQI_MD_FORMAT	Output	Contains one of the following formats: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2 <i>See note at the beginning of this table.</i>
IMFMQI_MD_PRIORITY	Output	Contains a decimal number in the range 0 to 999999999.
IMFMQI_MD_PERSISTENCE	Output	Contains one of the following: MQPER_PERSISTENT MQPER_NOT_PERSISTENT
IMFMQI_MD_MSGID	Input/ Output	Processes or receives up to a 32-byte MsgId. Processed exactly as received for input to, and as output from, the GET. No conversion of hexadecimal data is done in either case. For input, the value MQMI_NONE is valid.

Variable Name	Source	Notes
IMFMQI_MD_CORRELID	Input/ Output	Processes or receives up to a 32-byte CORRELID. Processed exactly as received for input to and as output from the GET. No conversion of hexadecimal data is done in either case. For input, the following constant values is recognized and converted to their corresponding hexadecimal equivalent: MQCI_NONE MQCI_NEW_SESSION
IMFMQI_MD_BACKOUTCOUNT	Output	Contains a decimal value in the range 0 to 255.
IMFMQI_MD_REPLYTOQ	Output	Contains up to a 48-character name of the ReplyToQ.
IMFMQI_MD_REPLYTOQMGR	Output	Contains up to a 48-character name of the ReplyToQMGR.
IMFMQI_MD_USERIDENTIFIER	Output	Contains up to a 12-character UserIdentifier.
IMFMQI_MD_ACCOUNTINGTOKEN	Output	Contains MQACT_NONE or up to a 32-byte AccountingToken. Processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_MD_APPLIDENTITYDATA	Output	Contains up to a 32-character value for ApplIdentityData.

Variable Name	Source	Notes
IMFMQI_MD_PUTAPPLTYPE	Output	<p>Contains a user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> MQAT_AIX MQAT_BROKER MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DEFAULT MQAT_DOS MQAT_DQM MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_JAVA MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNKNOWN MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_PUTAPPLNAME	Output	Contains up to a 28-character value for PutApplName.
IMFMQI_MD_PUTDATE	Output	Contains an 8-character date stamp.
IMFMQI_MD_PUTTIME	Output	Contains an 8-character time stamp.
IMFMQI_MD_APPLORIGINDATA	Output	Contains a 4-character value for ApplOriginData.
IMFMQI_MD_GROUPID	Output	If a MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GROUPID. Processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_MD_MSGSEQNUMBER	Output	If a MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 1 to 999999999.
IMFMQI_MD_OFFSET	Output	If a MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 0 to 999999999.

Variable Name	Source	Notes
IMFMQI_MD_MSGFLAGS	Output	<p>If a MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks:</p> <p>MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MD_ORIGINALLENGTH	Output	<p>Contains a decimal number, in the range 1 to 999999999, or MQOL_UNDEFINED.</p> <p><i>See note at the beginning of this table.</i></p>

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)"          /* Connect to the queue manager
*/
IMFMQI_OD_OBJECTNAME = JOHNB.QUEUE4    /* Set queue name                */
"IMFEXECMQI OPENOPTS(MQOO_BROWSE)"     /* Open the queue               */
IMFMQI_GMO_OPTIONS= 'MQGMO_BROWSE_NEXT' /* Browse msg on queue          */
"IMFEXEC MQI GET BUFFER(GETDATA)"
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"   /* Close the queue              */
"IMFEXEC MQI DISC"                     /* Disconnect from queue manager */
EXIT
```

This example highlights GETting a message from a local queue. The following describes the variable values that are in effect during the GET processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.

MQI OPEN

This statement opens a queue.

Command	Parameters
IMFEXEC MQI OPEN	HCONN(hconn) HOBJ(hobj) OOPTS(options)

The following table describes the IMFEXEC MQI OPEN parameters.

Parameter	Function	Notes
<p>Note: The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, OOPTS('MQOO_OUTPUT MQOO_SET_ALL_CONTEXT')</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter:</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier CONN command • Is not required <p>If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>
HOBJ	Provides the object handle that represents the object being opened.	<p>This parameter:</p> <ul style="list-style-type: none"> • Specifies a variable to be created to contain the object handle. • Is not required. <p>If omitted, the object handle value is placed into the default variable IMFHOBJ. The object handle is returned by the IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable.</p>

Parameter	Function	Notes
OOPTS	Specifies the options that control the OPEN action.	<p>This parameter is not required.</p> <p>If omitted, the OPEN options value is taken from the default variable IMFMQI_OO_OPTIONS. Either the parameter or the variable is required.</p> <p>Options can be specified as any valid combination of the following:</p> <ul style="list-style-type: none"> MQOO_BIND_NOT_FIXED MQOO_BIND_ON_OPEN MQOO_FAIL_IF QUIESCING MQOO_ALTERNATE_USER_AUTHORITY MQOO_SET_ALL_CONTEXT MQOO_SET_IDENTITY_CONTEXT MQOO_PASS_ALL_CONTEXT MQOO_PASS_IDENTITY_CONTEXT MQOO_SAVE_ALL_CONTEXT MQOO_INQUIRE MQOO_OUTPUT MQOO_BROWSE MQOO_INPUT_EXCLUSIVE MQOO_INPUT_SHARED MQOO_INPUT_AS_Q_DEF MQOO_BIND_AS_Q_DEF <p>See note at the beginning of this table.</p>

This table describes the variables used to build the object descriptor prior to issuing the OPEN statement.

Variable Name	Source	Notes
<p>Note: The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, OOPTS('MQOO_OUTPUT MQOO_SET_ALL_CONTEXT')</p>		
IMFMQI_OD_STRUCID	Input	(optional) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used unmodified by the EXEC Manager.
IMFMQI_OD_VERSION	Input	(optional) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTTYPE	Input	(optional) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTNAME	Input	Contains the queue name to be opened. Up to 48 characters can be specified.

Variable Name	Source	Notes
IMFMQI_OD_OBJECTQMGRNAME	Input	(<i>optional</i>) Contains the 48-character object queue manager name. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_DYNAMICQNAME	Input	(<i>optional</i>) Contains the 48-character dynamic queue name when using a model queue. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_ALTERNATEUSERID	Input	(<i>optional</i>) Contains the 12-character alternate user identifier. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_RECSPRESENT	Input	(<i>optional</i>) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_OD_KNOWNDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_UNKNOWNDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_INVALIDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_OBJECTRECOFFSET	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECOFFSET	Input	This variable contains 0.
IMFMQI_OD_OBJECTRECPtr	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECPtr	Input	This variable contains 0.
IMFMQI_OD_ALTERNATESECURITYID	Input	This variable exists but is null.
IMFMQI_OD_RESOLVEDQNAME	Output	This variable contains blanks.
IMFMQI_OD_RESOLVEDQMGRNAME	Output	This variable contains blanks.
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures used for this statement. For this statement, the variable contains 'OD'.
IMFMQI_OFFSETS	Output	Contains the value 'NONE NONE'. The variable is provided for consistency with other IMFEXEC MQI statements that use this variable and is not of significant use for the IMFEXEC MQI OPEN statement.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"/ * Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = TEST.QUEUE1/* Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"/ * Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT'/* Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECB1'
"IMFEXEC MQI PUT BUFFER(PUTDAT)" /* Put message on queue*/
"IMFEXEC MQI COMMIT" /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"/ * Close the queue */
"IMFEXEC MQI DISC" /* Disconnect from queue manager */
EXIT
```

This example highlights connecting to the queue manager whose connection handle will be stored in variable IMFHCONN.

- Variable IMFHCONN contains the connection handle name.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQI_REASON contains the constant (character) reason code.

MQI PUT

This statement puts a message to a previously opened queue.

Command	Parameters
IMFEXEC MQI PUT	HCONN(hconn) HOBJ(hobj) POPTS(options) BUFFLEN(bufferlength) BUFFER(buffer)

The following table describes the IMFEXEC MQI PUT parameters.

Parameter	Function	Notes
<p>Note: The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')</p>		
HCONN	Provides the connection handle that represents the connection to the queue manager.	<p>This parameter:</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier CONN command • Is not required <p>If omitted:</p> <ul style="list-style-type: none"> • The connection handle value is the default variable IMFHCONN. • The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. <p>Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.</p>
HOBJ	Provides the object handle that represents the queue to which the message is being PUT.	<p>This parameter:</p> <ul style="list-style-type: none"> • Specifies a variable created in an earlier OPEN command • Is not required <p>If omitted:</p> <ul style="list-style-type: none"> • The object handle value is taken from the default variable IMFHOBJ. • The object handle is returned by a previous IMFEXEC MQI OPEN request and stored in the IMFHOBJ variable or a user-specified variable. <p>Either this parameter, specifying an existing variable that contains the object handle, or a previously created IMFHOBJ variable containing the object handle is required.</p>

Parameter	Function	Notes
POPTS	Specifies the options that control the PUT action.	<p>This parameter is not required. If omitted, the PUT options value is taken from the default variable IMFMQI_PMO_OPTIONS. The variable IMFMQI_PMO_OPTIONS is not required. Options can be specified as any valid combination of the following choices:</p> <ul style="list-style-type: none"> MQPMO_SYNCPOINT MQPMO_NO_SYNCPOINT MQPMO_NO_CONTEXT MQPMO_DEFAULT_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT MQPMO_PASS_ALL_CONTEXT MQPMO_SET_IDENTITY_CONTEXT MQPMO_SET_ALL_CONTEXT MQPMO_ALTERNATE_USER_ AUTHORITY MQPMO_FAIL_IF QUIESCING MQPMO_NONE (IBM default) <p><i>See note at the beginning of this table.</i></p> <p>MAINVIEW AutoOPERATOR provides you with another option, REMOVE_DLH, in addition to the above options. With the REMOVE_DLH option, you can remove the Dead Letter Header from a message before putting it to another queue. The REMOVE_DLH is added to the POPTS() keyword parameter of the IMFEXEC MQI PUT statement. This is a MAINVIEW AutoOPERATOR option that causes the PUT message to be rebuilt by reloading the Encoding, CodedCharSetId and Format fields from the Dead Letter Header (DLH) into the Message Descriptor. This occurs before the PUT statement executes. The DLH, if present, is removed from the buffer containing the data.</p>
BUFFER	Specifies the variable that contains the message buffer data before issuing the PUT.	<p>This parameter is not required. If omitted, the variable IMFMQI_BUFFER must contain the message buffer data in order to complete the PUT. Be aware that upon successful completion of the PUT statement, the original buffer variable may be different if you have specified structure variables prior to the PUT. For example, you may want to build a Dead Letter message by changing the format variable,</p> <pre>IMFMQI_MD_FORMAT to MQFMT_DEAD_LETTER_HEADER</pre> <p>and by setting variables starting with IMFMQI_DLH_ and ending with Dead Letter (DLH) field names (see Appendix D for lists of structure variables). In this example, MAINVIEW AutoOPERATOR updates the buffer from the variables set before issuing the MQSeries PUT (see the section entitled "Creating Messages from Variables" on page 9-6 to learn how MAINVIEW AutoOPERATOR creates messages out of variables). The Dead Letter header is prefixed in front of the data that is in the buffer. MAINVIEW AutoOPERATOR updates the variable specified for the buffer so that it contains the DLH structure before control is returned to the EXEC.</p>

Parameter	Function	Notes
BUFFLEN	Specifies the length of the data used to create the variable containing the message buffer data.	<p>This parameter:</p> <ul style="list-style-type: none"> • Is not required. • Can specify either a numeric value or the name of a variable that contains a numeric value. • Can specify a value in the range of 0 to 32767, either specifically or using variable. <p>If this parameter is omitted, the variable IMFMQI_BUFFLEN will be used to obtain the BUFFER length. If neither is set, the length of the data in the message buffer variable is used.</p>

Table 9-3 describes the most common input and output variables for the IMFEXEC MQI PUT statement. Any required variables must be set prior to the PUT. For structures other than the MD (message descriptor), refer to Appendix D, “EXECs Variables” on page 1. Variables used begin with “IMFMQI_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore. For definitions of field names and constants used in this table that are derived from MQSeries structure field names, see the *IBM MQSeries Application Programming Reference Guide*.

Table 9-3 Common Input and Output Variables for the IMFEXEC MQI PUT Statement (Part 1 of 6)

Variable Name	Source	Notes
<p>Note: The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')</p>		
IMFMQI_MD_STRUCID	Input	(<i>optional</i>) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_MD_VERSION	Input	(<i>optional</i>) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used, unmodified by the EXEC Manager.
IMFMQI_MD_REPORT	Input	<p>(<i>optional</i>) Specify one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE (IBM default)</p> <p>See note at the beginning of this table.</p>
IMFMQI_MD_MSGTYPE	Input	<p>(<i>optional</i>) Specify one of the following:</p> <p>MQMT_DATAGRAM (IBM default) MQMT_REQUEST MQMT_REPLY MQMT_REPORT</p> <p>See note at the beginning of this table.</p>
IMFMQI_MD_EXPIRY	Input	(<i>optional</i>) Specify a decimal number, in the range 1 to 99999999, or MQEI_UNLIMITED (IBM default).

Table 9-3 Common Input and Output Variables for the IMFEXEC MQI PUT Statement (Part 2 of 6)

Variable Name	Source	Notes
IMFMQI_MD_FEEDBACK	Input	<p>(optional) Specify one of the following values for report messages:</p> <p>MQFB_NONE (IBM default) MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT</p> <p>See note at the beginning of this table.</p> <p>In addition, this variable can contain any reason code from the following sources:</p> <p>IMS-bridge feedback codes CICS-bridge feedback codes MQSeries reason codes User-specified numeric codes</p>
IMFMQI_MD_ENCODING	Input	<p>(optional) Specify MQENC_NATIVE (IBM default) or any decimal number up to 999999999.</p>
IMFMQI_MD_CODEDCHARS ETID	Input	<p>(optional) Contains one of the following:</p> <p>MQCCSI_Q_MGR MQCCSI_EMBEDDED Or any decimal number up to 999999999.</p> <p>See note at the beginning of this table.</p>
IMFMQI_MD_FORMAT	Input	<p>(optional) Specify one of the following:</p> <p>MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2</p> <p>See note at the beginning of this table</p>
IMFMQI_MD_PRIORITY	Input	<p>(optional) Specify a decimal number in the range 0 to 999999999 or MQPRI_PRIORITY_AS_Q_DEF (IBM default).</p>

Table 9-3 Common Input and Output Variables for the IMFEXEC MQI PUT Statement (Part 3 of 6)

Variable Name	Source	Notes
IMFMQI_MD_PERSISTENCE	Input	(<i>optional</i>) Specify one of the following: MQPER_PERSISTENT MQPER_NOT_PERSISTENT MQPER_PERSISTENCE_AS_Q_DEF (IBM default) <i>See note at the beginning of this table.</i>
IMFMQI_MD_MSGID	Input/ Output	(<i>optional</i>) Specify up to a 32-byte MsgId. Processed exactly as received. No conversion of hexadecimal data is done. For input, the value MQMI_NONE is valid. After a successful PUT, the variable is set from the field in the message descriptor.
IMFMQI_MD_CORRELID	Input/ Output	(<i>optional</i>) Specify up to a 32-byte CORRELID. Processed exactly as received. No conversion of hexadecimal data is done. After a successful PUT, the variable is set from the field in the message descriptor. For input, the following values are valid: MQCI_NONE (IBM default) MQCI_NEW_SESSION
IMFMQI_MD_BACKOUTCOUNT	N/A	This variable is not used for PUT.
IMFMQI_MD_REPLYTOQ	Input	(<i>optional</i>) Specify up to a 48-character name of the ReplyToQ. The IBM default value is 48 blanks.
IMFMQI_MD_REPLYTOQMGR	Input	(<i>optional</i>) Specify up to a 48-character name of the ReplyToQMGR. The IBM default value is 48 blanks.
IMFMQI_MD_USERIDENTIFIER	Input/ Output	(<i>optional</i>) Specify up to a 12-character UserIdentifier. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the UserIdentifier field is used as input and output.
IMFMQI_MD_ACCOUNTINGTOKEN	Output	(<i>optional</i>) Contains MQACT_NONE (IBM default) or up to a 32-byte AccountingToken. Processed exactly as received. No conversion of hexadecimal data is done.
IMFMQI_MD_APPLIDENTITYDATA	Input/ Output	(<i>optional</i>) Contains up to a 32-character value for ApplIdentityData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplIdentityData field is used as input and output.
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures used in the creation of the message. For example, in the case of a Dead Letter message, the contents would contain at least, 'MD DLH'.

Table 9-3 Common Input and Output Variables for the IMFEXEC MQI PUT Statement (Part 4 of 6)

Variable Name	Source	Notes
IMFMQI_OFFSETS	Output	<p>Contains a blank-delimited list of the offsets of the structures used in the creation of the message (as well as the offset of the application data) from the beginning of the message buffer. Using this variable you can access application data beyond the structures in the message buffer after issuing the PUT. The number of entries in the list is always one more than the number of structures participating in the operation, including MD, which participates in the operation but is not part of the message buffer. This is because an entry for the offset of the application data is included in the list. The value 'NONE' is always used for a structure that does not exist in the buffer but is used in creating the message for PUT, such as 'MD'. For example, if the message was a dead letter message and contained application data, the variable may look like this:</p> <p>'NONE 0 172'</p> <p>In the above example, NONE is specified for the MD, 0 for the offset of the Dead Letter Header and 172 is the offset of the application data.</p>

Table 9-3 Common Input and Output Variables for the IMFEXEC MQI PUT Statement (Part 5 of 6)

Variable Name	Source	Notes
IMFMQI_MD_PUTAPPLTYPE	Input/ Output	<p>(<i>optional</i>) Specify one of the following standard types or a numeric user-defined type:</p> <p>MQAT_AIX MQAT_BROKER MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DEFAULT MQAT_DOS MQAT_DQM MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_JAVA MQAT_MVS MQAT_NO_CONTEXT MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_UNKNOWN MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF</p> <p><i>See note at the beginning of this table.</i> To determine the circumstances under which the PutApplType field is used as input and output see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>
IMFMQI_MD_PUTAPPLNAME	Input/ Output	<p>(<i>optional</i>) Specify up to a 28-character value for PutApplName. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutApplName field is used as input and output.</p>
IMFMQI_MD_PUTDATE	Input/ Output	<p>(<i>optional</i>) Specify an 8-character date stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutDate field is used as input and output.</p>
IMFMQI_MD_PUTTIME	Input/ Output	<p>(<i>optional</i>) Specify an 8-character time stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutTime field is used as input and output.</p>

Table 9-3 Common Input and Output Variables for the IMFEXEC MQI PUT Statement (Part 6 of 6)

Variable Name	Source	Notes
IMFMQI_MD_APPLORIGINDATA	Input/ Output	(optional) Specify a 4-character value for ApplOriginData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplOriginData field is used as input and output.
IMFMQI_MD_GROUPID	Input/ Output	(optional) If the current MD is a MQMD_VERSION_2 MD, this variable may be set to a 24-byte GROUPID. Processed exactly as received from the PUT. No conversion of hexadecimal data is done. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the GROUPID field is used as input and output.
IMFMQI_MD_MSGSEQNUMBER	Input/ Output	(optional) If a MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 1 to 999999999 (IBM default 1). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the MsgSeqNumber field is used as input and output.
IMFMQI_MD_OFFSET	Input/ Output	(optional) If a MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 0 to 999999999 (IBM default 0). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the Offset field is used as input and output.
IMFMQI_MD_MSGFLAGS	Input	(optional) If a MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks: MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE (IBM default) See note at the beginning of this table.
IMFMQI_MD_ORIGINALLENGTH	Input	(optional) Contains a decimal number, in the range 1 to 999999999, or MQOL_UNDEFINED.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSQ1)"/** Connect to the queue manager */
IMFMQI_OD_OBJECTNAME = "TEST.QUEUE1"/** Set queue name */
"IMFEXEC MQI OPEN OOPTS(MQOO_OUTPUT)"/** Open the queue */
IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT'/** Set Put options*/
PUTDATA = 'MQPUT001 - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT BUFFER(PUTDATA)" /* Put message on queue*/
"IMFEXEC MQI COMMIT" /* Commit the work */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"/** Close the queue */
"IMFEXEC MQI DISC" /* Disconnect from queue manager */
EXIT
```

This example highlights PUTting a message to a local queue. The following describes the variable values that are in effect during the PUT processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.

MQI PUT1

This statement puts a message to a queue without the need for an OPEN and CLOSE.

Command	Parameters
IMFEXEC MQI PUT1	HCONN(hconn) POPTS(options) BUFFLEN(bufferlength) BUFFER(buffer)

The following table describes the IMFEXEC MQI PUT1 parameters.

Parameter	Function	Notes
Note: The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .		
HCONN	Provides the connection handle that represents the connection to the queue manager.	This parameter: <ul style="list-style-type: none"> • Specifies a variable created in an earlier CONN command • Is not required If omitted, the connection handle value is the default variable IMFHCONN. The connection handle is returned by a previous IMFEXEC MQI CONN request and stored in the IMFHCONN variable or a user-specified variable. Either this parameter, specifying an existing variable that contains the connection handle, or a previously created IMFHCONN variable containing the connection handle is required.

Parameter	Function	Notes
POPTS	Specifies the options that control the PUT1 action.	<p>This parameter is not required. If omitted, the PUT1 options are taken from the default variable IMFMQI_PMO_OPTIONS. The variable IMFMQI_PMO_OPTIONS is not required. Valid PUT1 options:</p> <ul style="list-style-type: none"> MQPMO_SYNCPOINT MQPMO_NO_SYNCPOINT MQPMO_NO_CONTEXT MQPMO_DEFAULT_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT MQPMO_PASS_ALL_CONTEXT MQPMO_SET_IDENTITY_CONTEXT MQPMO_SET_ALL_CONTEXT MQPMO_ALTERNATE_USER_AUTHORITY MQPMO_FAIL_IF QUIESCING MQPMO_NONE (IBM default) <p><i>See note at the beginning of this table.</i></p> <p>When specifying multiple options, they must be enclosed within single quotation marks, for example: POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')</p> <p>In addition to the above options, another MAINVIEW AutoOPERATOR-provided option (REMOVE_DLH) is available.</p> <p>Using the REMOVE_DLH option, you can remove the Dead Letter Header from a message before PUTting it to another queue. REMOVE_DLH is added to the POPTS() keyword parameter of the IMFEXEC MQI PUT1 statement. This MAINVIEW AutoOPERATOR option causes the message being PUT1 to be rebuilt by reloading the Encoding, CodedCharSetId and Format fields from the Dead Letter Header (DLH) into the Message Descriptor. This occurs before the PUT1 statement executes. The DLH, if present, is removed from the buffer containing the data.</p>
BUFFER	Specifies the variable that contains the message buffer data prior to issuing a PUT1.	<p>This parameter is not required. If omitted, the variable IMFMQI_BUFFER must contain the message buffer data in order to complete the PUT1. Be aware that upon successful completion of the PUT1 statement, the original buffer variable may be different if you have specified structure variables prior to the PUT1. For example, you may want to build a Dead Letter message by changing the format variable, IMFMQI_MD_FORMAT, to MQFMT_DEAD_LETTER_HEADER and by setting variables starting with IMFMQI_DLH_ and ending with Dead Letter (DLH) field names (see Appendix D for lists of structure variables). In this example, MAINVIEW AutoOPERATOR updates the buffer from the variables set before issuing the MQSeries PUT1 (see the section entitled "Creating Messages from Variables" on page 9-6 to learn how MAINVIEW AutoOPERATOR creates message out of variables). The Dead Letter header is prefixed in front of the data that is currently in the buffer. MAINVIEW AutoOPERATOR updates the variable specified for the buffer so that it contains the DLH structure before control is returned to the EXEC.</p>

Parameter	Function	Notes
BUFFLEN	Specifies the length of the data used to create the message buffer data.	<p>This parameter</p> <ul style="list-style-type: none"> • Is not required • Can specify a numeric value or the name of a variable that contains a numeric value • Can specify a value in the range of 0 to 32767, either specifically, or using a variable <p>If this parameter is omitted, the variable IMFMQI_BUFFLEN is used to obtain the BUFFER length. If neither is set, the length of the data in the message buffer variable is used.</p>

The following table describes the most common input and output variables for the IMFEXEC MQI PUT1 statement. Any required variables must be set prior to the PUT. For structures other than the MD (message descriptor) and the OD (object descriptor), please refer to Appendix D, “EXECs Variables” on page 1. Variables used begin with “IMFMQI_”, followed by the MQSeries structure, followed by the field name within the MQSeries structure, with each node delimited by an underscore. For definitions of field names and constants derived from MQSeries structure field names and used in this table, see the *IBM MQSeries Application Programming Reference Guide*.

Table 9-4 Variables and Source (Part 1 of 8)

Variable Name	Source	Notes
<p>Note: The following list of constants may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>. Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, POPTS('MQPMO_NO_SYNCPOINT MQPMO_SET_ALL_CONTEXT')</p>		
IMFMQI_OD_STRUCID	Input	(<i>optional</i>) The EXEC may or may not set this variable. If the EXEC does not set this variable, the MQSeries default value will be used unmodified by the EXEC Manager.
IMFMQI_OD_VERSION	Input	(<i>optional</i>) The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTTYPE	Input	(<i>optional</i>) The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_OD_OBJECTNAME	Input	Contains the queue name to be opened; up to 48 characters can be specified.
IMFMQI_OD_OBJECTQMGRNAME	Input	(<i>optional</i>) Contains the 48-character object queue manager name. The EXEC Manager does not specify any default value when this variable is not set.

Table 9-4 Variables and Source (Part 2 of 8)

Variable Name	Source	Notes
IMFMQI_OD_DYNAMICQNAME	Input	<i>(optional)</i> Contains the 48-character dynamic queue name when using a model queue. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_ALTERNATEUSERID	Input	<i>(optional)</i> Contains the 12-character alternate user identifier. The EXEC Manager does not specify any default value when this variable is not set.
IMFMQI_OD_RECSPRESENT	Input	<i>(optional)</i> The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_OD_KNOWNDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_UNKNOWNDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_INVALIDDESTCOUNT	Output	This variable contains 0.
IMFMQI_OD_OBJECTRECOFFSET	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECOFFSET	Input	This variable contains 0.
IMFMQI_OD_OBJECTRECPtr	Input	This variable contains 0.
IMFMQI_OD_RESPONSERECPtr	Input	This variable contains 0.
IMFMQI_OD_ALTERNATESECURITYID	Input	This variable exists but is null.
IMFMQI_OD_RESOLVEDQNAME	Output	This variable contains blanks.
IMFMQI_OD_RESOLVEDQMGRNAME	Output	This variable contains blanks.
IMFMQI_MD_STRUCID	Input	<i>(optional)</i> The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.
IMFMQI_MD_VERSION	Input	<i>(optional)</i> The EXEC may or may not set this variable. If the EXEC does not set this value, the MQSeries default will be used, unmodified by the EXEC Manager.

Table 9-4 Variables and Source (Part 3 of 8)

Variable Name	Source	Notes
IMFMQI_MD_REPORT	Input	<p>(<i>optional</i>) Specify one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE (IBM default)</p> <p>See note at the beginning of this table.</p>
IMFMQI_MD_MSGTYPE	Input	<p>(<i>optional</i>) Specify one of the following:</p> <p>MQMT_DATAGRAM (IBM default) MQMT_REQUEST MQMT_REPLY MQMT_REPORT</p> <p>See note at the beginning of this table.</p>
IMFMQI_MD_EXPIRY	Input	<p>(<i>optional</i>) Specify a decimal number, in the range 1 to 999999999, or MQEI_UNLIMITED (IBM default).</p>

Table 9-4 Variables and Source (Part 4 of 8)

Variable Name	Source	Notes
IMFMQI_MD_FEEDBACK	Input	<p>(<i>optional</i>) Specify one of the following values for report messages:</p> <p>MQFB_NONE (IBM default) MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT</p> <hr/> <p>See note at the beginning of this table. In addition, this variable can contain any reason code from the following sources:</p> <p>IMS-bridge feedback codes CICS-bridge feedback codes MQSeries reason codes User-specified numeric codes</p>
IMFMQI_MD_ENCODING	Input	<p>(<i>optional</i>) Specify one of the following: MQENC_NATIVE (IBM default) or any decimal number up to 999999999.</p>
IMFMQI_MD_CODEDCHARSETID	Input	<p>(<i>optional</i>) Contains one of the following: MQCCSI_Q_MGR MQCCSI_EMBEDDED Or any decimal number up to 999999999.</p> <p>See note at the beginning of this table.</p>
IMFMQI_MD_FORMAT	Input	<p>(<i>optional</i>) Specify one of the following:</p> <p>MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2</p> <p>See note at the beginning of this table.</p>

Table 9-4 Variables and Source (Part 5 of 8)

Variable Name	Source	Notes
IMFMQI_MD_PRIORITY	Input	(<i>optional</i>) Specify a decimal number in the range 0 to 999999999 or MQPRI_PRIORITY_AS_Q_DEF (IBM default).
IMFMQI_MD_PERSISTENCE	Input	(<i>optional</i>) Specify one of the following: MQPER_PERSISTENT MQPER_NOT_PERSISTENT MQPER_PERSISTENCE_AS_Q_DEF (IBM default) See note at the beginning of this table.
IMFMQI_MD_MSGID	Input/ Output	(<i>optional</i>) Specify up to a 32-byte MsgId. Processed exactly as received. No conversion of hexadecimal data is done. For input, the value MQMI_NONE is valid. After a successful PUT, the variable is set from the field in the message descriptor.
IMFMQI_MD_CORRELID	Input/ Output	(<i>optional</i>) Specify up to a 32-byte CORRELID. Processed exactly as received. No conversion of hexadecimal data is done. After a successful PUT, the variable is set from the field in the message descriptor. For input, the following values are valid: MQCI_NONE (IBM default) MQCI_NEW_SESSION
IMFMQI_MD_BACKOUTCOUNT	N/A	This variable is not used for PUT.
IMFMQI_MD_REPLYTOQ	Input	(<i>optional</i>) Specify up to a 48-character name of the ReplyToQ. The IBM default value is 48 blanks.
IMFMQI_MD_REPLYTOQMGR	Input	(<i>optional</i>) Specify up to a 48-character name of the ReplyToQMGR. The IBM default value is 48 blanks.
IMFMQI_MD_USERIDENTIFIER	Input/ Output	(<i>optional</i>) Specify up to a 12-character UserIdentifier. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the UserIdentifier field is used as input and output.
IMFMQI_MD_ACCOUNTINGTOKEN	Output	(<i>optional</i>) Contains MQACT_NONE (IBM default) or up to a 32-byte AccountingToken. Processed exactly as received. No conversion of hexadecimal data is done.
IMFMQI_MD_APPLIDENTITYDATA	Input/ Output	(<i>optional</i>) Contains up to a 32-character value for ApplIdentityData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplIdentityData field is used as input and output.

Table 9-4 Variables and Source (Part 6 of 8)

Variable Name	Source	Notes
IMFMQI_MD_PUTAPPLTYPE	Input/ Output	<p>(optional) Specify one of the following standard types or a numeric user-defined type:</p> <p>MQAT_AIX MQAT_BROKER MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DEFAULT MQAT_DOS MQAT_DQM MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_JAVA MQAT_MVS MQAT_NO_CONTEXT MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNKNOWN MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF</p> <p>See note at the beginning of this table. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutApplType field is used as input and output.</p>
IMFMQI_MD_PUTAPPLNAME	Input/ Output	<p>(optional) Specify up to a 28-character value for PutApplName. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutApplName field is used as input and output.</p>
IMFMQI_MD_PUTDATE	Input/ Output	<p>(optional) Specify an 8-character date stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutDate field is used as input and output.</p>
IMFMQI_MD_PUTTIME	Input/ Output	<p>(optional) Specify an 8-character time stamp. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the PutTime field is used as input and output.</p>

Table 9-4 Variables and Source (Part 7 of 8)

Variable Name	Source	Notes
IMFMQI_MD_APPLORIGINDATA	Input/ Output	(<i>optional</i>) Specify a 4-character value for ApplOriginData. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the ApplOriginData field is used as input and output.
IMFMQI_MD_GROUPID	Input/ Output	(<i>optional</i>) If the current MD is an MQMD_VERSION_2 MD, this variable may be set to a 24-byte GROUPID. Processed exactly as received from the PUT1. No conversion of hexadecimal data is done. See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the Grouped field is used as input and output.
IMFMQI_MD_MSGSEQNUMBER	Input/ Output	(<i>optional</i>) If a MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 1 to 999999999 (IBM default 1). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the MsgSeqNumber field is used as input and output.
IMFMQI_MD_OFFSET	Input/ Output	(<i>optional</i>) If a MQMD_VERSION_2 MD is present, this variable may contain a decimal number in the range 0 to 999999999 (IBM default 0). See the <i>IBM MQSeries Application Programming Reference Guide</i> to determine the circumstances under which the Offset field is used as input and output.
IMFMQI_MD_MSGFLAGS	Input	(<i>optional</i>) If a MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks: MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE (IBM default) <i>See note at the beginning of this table.</i>
IMFMQI_MD_ORIGINALLENGTH	Input	(<i>optional</i>) Contains a decimal number, in the range 1 to 999999999, or MQOL_UNDEFINED.

Table 9-4 Variables and Source (Part 8 of 8)

Variable Name	Source	Notes
IMFMQI_STRUCTURES	Output	Contains a blank-delimited list of the MQSeries structures used in the creation of the message. For example, in the case of a Dead Letter message, the contents would contain at least, 'MD OD DLH'.
IMFMQI_OFFSETS	Output	Contains a blank-delimited list of the offsets of the structures used in the creation of the message (as well as the offset of the application data) from the beginning of the message buffer. Using this variable you can access application data beyond the structures in the message buffer after issuing the PUT1. The number of entries in the list is always one more than the number of structures participating in the operation, including MD, which participates in the operation but is not part of the message buffer. This is because an entry for the offset of the application data is included in the list. The value 'NONE' is always used for a structure that does not exist in the buffer but is used in creating the message for PUT1, such as 'MD'. For example, if the message was a dead letter message and contained application data, the variable may look like this: 'NONE NONE 0 172' In the above example, NONE is specified for the OD, 0 for the offset of the Dead Letter Header and 172 is the offset of the application data.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see "How Completion and Reason Codes Are Returned" on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```
/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)"/ * Connect to the queue manager      */
IMFMQI_OD_OBJECTNAME = "JOHNB.QUEUE1"/ * Set queue name          */
IMFMQI_PMO_OPTIONS = 'MQPMO_NO_SYNCPOINT'/* Set Put options      */
PUTDATA = 'MQEXAMPL - TEST DATA ECB = MQPUTECB1 POST CODE = PUTECEB1'
"IMFEXEC MQI PUT1 BUFFER(PUT DATA)"/ * Put message on queue    */
"IMFEXEC MQI DISC"/ * Disconnect from queue manager             */
EXIT
```

This example highlights the PUTting of a message to a local queue. The following describes the variable values that are in effect during the PUT processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.

CMD (Issue IMFEXEC Command to MQSeries with Response)

This format issues commands to the MQSeries command server. A response is returned to the issuing EXEC.

Command	Parameters
IMFEXEC CMD	'MQ command' TYPE(mqs) LM(queue manager name) RESPONSE(msgid1,msgid2,...) WAIT(30 nnnn) PCF(yes no) QM(queue manager name) CQ(command queue name)

The following table describes the parameters:

Parameter	Function	Notes
MQ command	A valid MQ command	This parameter <ul style="list-style-type: none"> • Is required • Has no default value The MQ command can be up to 256 characters long.
TYPE	Type of IMFEXEC CMD	This parameter <ul style="list-style-type: none"> • Is required • Uses a default of MVS You must code TYPE(mqs) in order to issue the command to a MQ command server.
LM	Local MQ queue manager name	This parameter <ul style="list-style-type: none"> • Is not required • Uses the default MQ manager name You can issue MQ commands to any MQ queue manager that is on the same OS/390 system as MAINVIEW AutoOPERATOR. If this parameter is not specified, the default queue manager will be used. (Refer to the IBM MQ documentation for information about how the default MQ queue manager is determined.)

Parameter	Function	Notes
RESPONSE	Allows you to specify from 1 to 8 message IDs	<p>This parameter</p> <ul style="list-style-type: none"> • Is not required • Has no default value <p>Each message ID may contain up to 16 characters including wildcard characters. An example message ID is 'CSQN205I'. If this parameter is coded, only command response messages that match at least one of the message IDs will be returned to the EXEC. Do not code this parameter if you want to receive all the command response messages.</p>
WAIT	A number from 5 to 9999 (in seconds)	<p>This parameter</p> <ul style="list-style-type: none"> • Is not required • Uses a default value of 30 <p>Number of seconds that the IMFEXEC CMD waits for MQ command response messages. MAINVIEW AutoOPERATOR for MQSeries need not always wait this number of seconds; in fact, the EXEC will continue processing as soon as all MQ command response messages have been received. Unless you are experiencing problems with a slow MQ command server, do not code this parameter.</p>
PCF	With this parameter, you can specify whether to issue commands in PCF format. Valid values are YES or NO.	<p>This parameter</p> <ul style="list-style-type: none"> • Is not required • Uses a default value of no <p>When PCF=YES, MAINVIEW AutoOPERATOR issues commands in PCF (Programmable Command Format) format. You need to code this parameter only when issuing commands to a non-OS/390 MQ queue manager.</p>
QM	With this parameter, you can specify the name of a MQ queue manager to which you want to send a command.	<p>This parameter</p> <ul style="list-style-type: none"> • Is not required • Has no default value • Supports 1 to 48 characters. This queue manager can be OS/390 or non-OS/390. <p><i>This field is case sensitive (others are automatically upper cased).</i></p> <p>MAINVIEW AutoOPERATOR issues the command through the local MQ queue manager specified (or defaulted) by the NAME() parameter.</p>
CQ	The name of the system command queue at the remote MQ queue manager	<p>This parameter:</p> <ul style="list-style-type: none"> • Is not required • Has no default value <p>Specify the name of the system command queue (1 to 48 characters) of the remote MQ queue manager. For example, the queue SYSTEM.ADMIN.COMMAND.QUEUE is used for many non-OS/390 MQ queue managers.</p>
<p>For more information about the parameters described in this table, refer to the <i>IBM MQSeries Application Programming Reference</i>.</p>		

In addition, note the following information:

- The number of MQ response messages is returned in the TSO variable IMFNOL.
- The TSO variables MQLNxxxx contain the data received (for example, MQLN1 contains the first response message, and MQLN2 contains the second, and so on). Up to 9999 response messages are supported. The maximum size of each MQLNxxxx variable (response message) is 13000 characters.
- The CSQ9022I message (for example, CSQ9022I @MQS1CPF CSQMDRTS 'DISPLAY QUEUE' NORMAL COMPLETION) contains embedded single quotation marks and non-displayable characters, which can cause problems when using IMFEXEC MSG and some other commands.

One solution to this problem is to translate the single quotation marks and invalid characters to other characters using the REXX TRANSLATE function. Refer to “Examples – Retrieving Responses to a MQ Command” on page 9-62 for an illustration.

- MAINVIEW AutoOPERATOR creates a permanent dynamic queue, 'BBOMVAO.EXEC.REPLY.aosid.qmgrssid'. This queue is modeled after the IBM-supplied SYSTEM.COMMAND.REPLY.MODEL, which is defined by CSQ4INP2 with the SHARE attribute. MAINVIEW AutoOPERATOR requires this Share attribute to allow multiple EXECs to concurrently retrieve response messages from MQ.
- The first OS/390 MQ command response message (CSQN205I) carries the count of the MQ command response messages as well as the return and reason codes for the command. Since the number of response messages is known, there is no need to specify it.
- MAINVIEW AutoOPERATOR destructively reads all MQ command responses (for example, the number of messages specified in CSQN205I) for this IMFEXEC CMD before returning control to the EXEC.
- If MQ detects an error with the command, IMFEXEC CMD fails with IMFCC=12. If you get this failure, retrieve all the MQ response messages and inspect the CSQN205I message carrying the MQ return and reason codes. In addition, inspect the MQ messages that follow the CSQN205I message. These messages provide more descriptions about the error.

- MAINVIEW AutoOPERATOR inserts the name of the user into the mqmd_UserIdentifier field of the message containing the command. It inserts the TS (Terminal Session) user for user initiated EXECs, and the SSID (PAS-ID) for automatically invoked (rule, timer, and so on) EXECs. You must ensure that this user ID has appropriate authority to issue the command at the destination MQ queue manager, particularly when it is a non-OS/390 MQ queue manager.
- MAINVIEW AutoOPERATOR issues remote commands through the local MQ queue manager specified (or defaulted) in the NAME() parameter. The remote MQ queue manager must be connected to this local MQ queue manager. In addition, *one* of the following requirements must be met at each MQ queue manager:
 - There is a queue manager alias to the other MQ queue manager.
 - There is a transmission queue with the name of the other MQ queue manager.

One of the following completion code is returned in the IMFCC variable:

Value	Description
0	Successful completion
4	Time out occurred while waiting for MQ command response messages. IMFNOL contains the number of response message received so far.
8	MQSeries returned error
12	Buffer too small
16	Syntax error
20	Variable Processing routine not found

Examples – Retrieving Responses to a MQ Command

Example 1 – Retrieving All Responses to a MQ Command

```

/* REXX */
/* Example to retrieve all responses to a MQ command*/
PARSE ARG EXNAME.
"IMFEXEC MSG '."EXNAME "EID="IMFEID "STARTED'"
"IMFEXEC CMD 'DISPLAY QUEUE(SYSTEM.CHANNEL.*)' TYPE(MQS)"
"IMFEXEC MSG '.IMFNOL="IMFNOL "CC="IMFCC "IMFEXEC CMD MQ'"
DO i=1 TO IMFNOL
bad_chars = xrange('\00'x, '\41'x);
bad_chars = bad_chars"""";
v1 = value('\MQLN'i);
clean_data = translate(v1, ' ', bad_chars)
  "IMFEXEC MSG '.MQLN"i "LENGTH=length(v1)""
  "IMFEXEC MSG '.MQLN"i "="clean_data""
END
"IMFEXEC MSG '."EXNAME "EID="IMFEID "CC="IMFCC "RC="IMFRC "ENDED'"

```

Example 2 – Retrieving Selected Responses to a MQ Command

```

/* REXX */
/* Example to retrieve selected responses to a MQ command.*/
PARSE ARG EXNAME.
"IMFEXEC MSG '."EXNAME "EID="IMFEID "STARTED'"
/* example of an IMFEXEC command that continues on the next line */
"IMFEXEC CMD 'DISPLAY QUEUE(SYSTEM.CHANNEL.*)' TYPE(MQS) ,
LM(MQS1) RESP(CSQN205I,CSQ9022I)"
"IMFEXEC MSG '.IMFNOL="IMFNOL "CC="IMFCC "IMFEXEC CMD MQ'"
DO i=1 TO IMFNOL
bad_chars = xrange('\00'x, '\41'x);
bad_chars = bad_chars"""";
v1 = value('\MQLN'i);
clean_data = translate(v1, ' ', bad_chars)
  "IMFEXEC MSG '.MQLN"i "LENGTH=length(v1)""
  "IMFEXEC MSG '.MQLN"i "="clean_data""
END
"IMFEXEC MSG '."EXNAME "EID="IMFEID "CC="IMFCC "RC="IMFRC "ENDED'"

```

COPY MQI

Using this statement, you can copy the contents of a set (or sets) of variables created from a structure (or multiple structures), found in a MQSeries message, to another set of variables for the purpose of temporarily saving. For example, suppose you want to GET a message from one queue, save the message descriptor values for later, GET another message and inspect some portion of that message and, based on values discovered from the second GET, forward the first message to another queue. Using the COPY MQI command, you can do this by saving the first set of structure values. Later, using another COPY command, you can copy them back to the IMFMQI_ variables and, optionally, PUT the message to another queue.

The IMFEXEC COPY MQI statement will copy entire structures based on input parameters. You can copy a single structure, for example, the MD, or multiple structures, for example, the MD and the MDE or ALL structures that exist in the message. You can also use the COPY MQI statement to nullify all variables in a structure or more than one structure.

Command	Parameters
IMFEXEC COPY MQI	STRTYPE(structure lds) STRFROM(input prefix) STRTO(output prefix)

This table describes the IMFEXEC COPY MQI parameters.

Parameters	Function	Notes
STRTYPE	Identifies the structure IDs to copy.	<p>This parameter is required. Possible input values are as follows:</p> <ul style="list-style-type: none"> MD (Message Descriptor) MDE (Message Descriptor Extension) OD (Object Descriptor) DLH (Dead Letter Header) CIH (CICS Information Header) IIH (IMS Information Header) XQH (Transmission Queue Header) TM (Trigger Monitor structure) TMC (Trigger Monitor structure Character format) WIH (Work Information Header) RMH (Reference Message Header) CFH (Command Format Header) RFH (Rules and Formatting Header) RFH2 (Rules and Formatting Header – version 2) EVEN (Event variables) ALL (all known existing structures) <p>Multiple structures can be specified, delimited by blanks or commas. The variables IMFMQI_STRUCTURES or IMFQ_STRUCTURES can be used in place of individual entries. The IMFMQI_STRUCTURES variable is returned from an IMFEXEC MQI GET statement and contains a blank-delimited list of structures that exist within the message. The IMFQ_STRUCTURES variable is passed to an EXEC from a MQSeries rule and contains a blank-delimited list of structures that exist within the message.</p> <p>Note: The variable IMFMQI_BUFFER (by default) contains the contents of the message buffer upon a successful GET. If this variable or another user-specified variable containing the message buffer exists during a COPY MQI operation, the EXEC may need to preserve the contents by issuing an assignment statement to set another variable to the value of the buffer variable prior to issuing another IMFEXEC MQI GET statement. For example, in REXX the statement might look like this: MYNEW_BUFFER = IMFMQI_BUFFER</p>

Parameters	Function	Notes
STRFROM	Specifies the prefix of the variables to copy from.	This parameter is not required. If omitted, the default variable prefix is IMFMQI_. The value specified must contain characters acceptable to CLIST and REXX. To set a structure of variables to nulls, specify STRFROM(NULLS). The allowable input prefix length is from 1 to 24 characters.
STRTO	Specifies the prefix of the variables to copy to	This parameter is required. The length of the value specified must be from 1 to 24 characters and must consist of characters acceptable to CLIST and REXX. Do not use STRTO value of BBIJRNL_ because MAINVIEW AutoOPERATOR reserves this for internal processing.

The completion code is returned in the IMFMQCC and IMFCC variables. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)" /* Connect to the queue manager*/
IMFMQI_OD_OBJECTNAME = 'JOHNB.QUEUE4' /* Set queue name*/
"IMFEXEC MQI OPEN OOPTS(MQOO_BROWSE)" /* Open the queue*/
IMFMQI_GMO_OPTIONS = 'MQGMO_BROWSE_NEXT' /* Set Get options*/
/* Browse msg on queue*/
"IMFEXEC MQI GET BUFFER(GETDATA)"
"IMFEXEC COPY MQI STRTYPE("IMFMQI_STRUCTURES")
                STRFROM(IMFMQI_) STRTO(IMFMQX_) /* Make a copy */
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)" /* Close the queue*/
"IMFEXEC MQI DISC"/* Disconnect from queue manager*/
EXIT

```

This example highlights how to copy a message to a local queue. The following variable values are in effect during the COPY processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.

DISPLAY MQI

Using this statement you can display the contents of a set (or sets) of variables created from a structure (or multiple structures) found in a MQSeries message, in the BBI journal. The variable contents are displayed in character format and in hexadecimal format. This information could be useful for debugging when variable comparisons are not matching and you are not able to determine why.

Command	Parameters
IMFEXEC DISPLAY MQI	STRTYPE(structure lds) PREFIX(input prefix)

The following table describes the IMFEXEC DISPLAY MQI parameters.

Parameters	Function	Notes
STRTYPE	Identifies the structure variables to display.	<p>This parameter is required.</p> <p>Possible input values are:</p> <ul style="list-style-type: none"> MD (Message Descriptor) MDE (Message Descriptor Extension) OD (Object Descriptor) DLH (Dead Letter Header) CIH (CICS Information Header) IIH (IMS Information Header) XQH (Transmission Queue Header) TM (Trigger Monitor structure) TMC (Trigger Monitor structure character format) WIH (Work Information Header) RMH (Reference Message Header) CFH (Command Format Header) RFH (Rules and Formatting Header) RFH2 (Rules and Formatting Header – version 2) EVEN (Event variables) ALL (all known existing structures) <p>Multiple structures can be specified, delimited by blanks or commas. The variables <code>IMFMQI_STRUCTURES</code> or <code>IMFQ_STRUCTURES</code> can be used in place of individual entries. The <code>IMFMQI_STRUCTURES</code> variable is returned from an IMFEXEC MQI GET statement and contains a blank-delimited list of structures that exist within the message. The <code>IMFQ_STRUCTURES</code> variable is passed to an EXEC from a MQSeries rule and contains a blank-delimited list of structures that exist within the message.</p>

Parameters	Function	Notes
PREFIX	Specifies the prefix of the variables to display.	This parameter is not required. If omitted, the default variable prefix is IMFMQI_.

The completion code is returned in the IMFCC variable. For additional information, see “How Completion and Reason Codes Are Returned” on page 9-5.

The reason code is returned in the IMFMQRC and IMFRC variables; contents of both variables are the same. Refer to the *IBM MQSeries Application Programming Reference Guide* for the reason code description.

Example

```

/* REXX */
"IMFEXEC MQI CONN NAME(CSBD)"           /* Connect to the queue manager
IMFMQI_OD_OBJECTNAME = "JOHNB.QUEUE4"   /* Set queue name*/
"IMFEXEC MQI OPEN OOPTS(MQOO_BROWSE)"    /* Open the queue*/
IMFMQI_GMO_OPTIONS = 'MQGMO_BROWSE_NEXT' /* Set Get options*/
/* Browse msg on queue*/
"IMFEXEC MQI GET BUFFER(GETDATA)"
"IMFEXEC DISPLAY MQI STRTYPE("IMFMQI_STRUCTURES")
      PREFIX(IMFMQI_)"
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE)"     /* Close the queue*/
"IMFEXEC MQI DISC"                       /* Disconnect from queue manager
EXIT

```

This example highlights the DISPLAYing of a message to a local queue. The following describes the variable values that are in effect during the DISPLY processing:

- Variable IMFHCONN contains the connection handle name.
- Variable IMFHOBJ contains the object handle for the queue.
- Variable IMFMQCC and IMFCC contain the completion code.
- Variable IMFMQRC and IMFRC contain the reason code.
- Variable IMFMQI_REASON contains the constant (character) reason code.

Appendix A Diagnosing MAINVIEW AutoOPERATOR for MQSeries Errors

This appendix provides resources and suggestions to help you resolve problems that you may encounter while installing or using MAINVIEW AutoOPERATOR for MQSeries.

Where You Can Find Additional Information

The following books contain additional information that you may find useful while resolving error messages or deciphering codes:

- *IBM MQSeries for OS/390 Messages and Codes*, SC33-0819

Use this book to look up messages or codes that you might receive from MQSeries.

- *IBM MQSeries Programmable System Management*, SC33-1482

Use this book to read about facilities available in MQSeries products for

- Monitoring instrumentation events in a network of connected systems that use IBM MQSeries products in different operating system environments
- Looking up descriptions of queue manager event messages

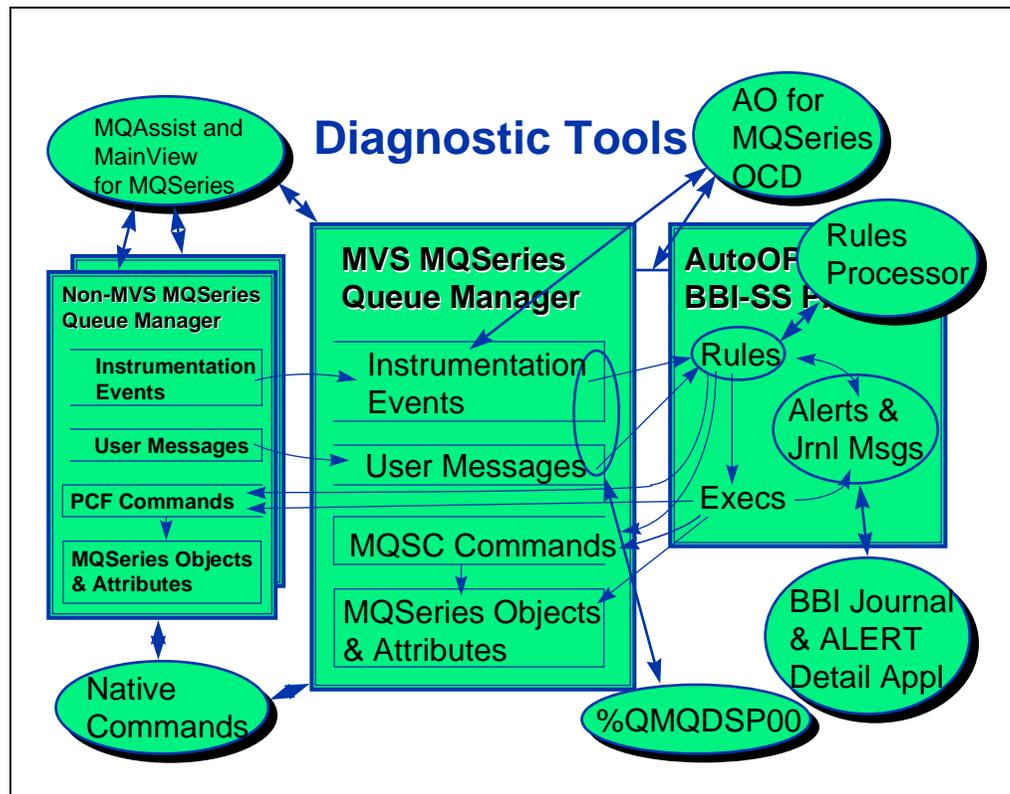
- *IBM MQSeries for OS/390 Problem Determination Guide, GC33-0808*

Use this book to help you determine the causes of MQSeries for OS/390 problems.

Which Tools Are Available to Diagnose MAINVIEW AutoOPERATOR for MQSeries Problems

Figure A-1 shows the interaction between MAINVIEW AutoOPERATOR for MQSeries and MQSeries and indicates the appropriate diagnostic tools to deal with each area.

Figure A-1 Interactions between MAINVIEW AutoOPERATOR for MQSeries and MQSeries Queue



Managers

This figure shows MAINVIEW AutoOPERATOR for MQSeries running in a BBI-SS PAS connected directly to an OS/390 MQSeries queue manager. The OS/390 queue manager is connected to non-OS/390 queue managers. For MAINVIEW AutoOPERATOR for MQSeries to detect the arrival of event messages and non-event messages from the non-OS/390 queue managers, the local event queues on the non-OS/390 queue managers have been deleted and redefined as remote queues that actually reside on the OS/390 queue manager. Therefore, MAINVIEW AutoOPERATOR for MQSeries can listen for both OS/390 queue manager events and non-OS/390 queue manager events.

Some of the important dynamics of Figure A-1 are as follows:

- All non-OS/390 instrumentation events must travel through an OS/390 queue manager to reach MAINVIEW AutoOPERATOR for MQSeries.
- OS/390 (and, indirectly non-OS/390) queue manager instrumentation events and non-event messages cause MAINVIEW AutoOPERATOR Rules to be fired. A Rule's actions can automate the event and additionally issue an ALERT to notify operators of important situations. A Rule can also schedule an EXEC to take additional automation actions.
- MAINVIEW AutoOPERATOR Rules and EXECs can issue MQSC commands for OS/390 queue managers (or PCF commands for non-OS/390 queue managers) to modify MQSeries objects and attributes.
- Both Rules and EXECs can write to the BBI Journal and the ALERT Detail application, which can then be referenced for verifying MAINVIEW AutoOPERATOR activities, such as firing a Rule.

Diagnostic Checklist

When MAINVIEW AutoOPERATOR for MQSeries automation does not appear to be functioning correctly, specific diagnostic tools are available to monitor particular areas. These are shown in the circled areas of Figure A-1 on page 3. The following checklist is the most systematic way to use the diagnostic tools to determine the solutions for MAINVIEW AutoOPERATOR for MQSeries problems:

1. Look up any messages or codes.

Examine the BBI Journal and the ALERT Detail application for error messages or other indications that MAINVIEW AutoOPERATOR is not working correctly. For example, if an expected action did not occur, the BBI Journal may display whether or not the Rule fired that would have caused the action.

Refer to the *IBM MQSeries for OS/390 Messages and Codes* for more information about specific OS/390 messages. Depending on the platform, refer to the appropriate messages and codes book for non-OS/390 messages.

2. Verify that QAO is active.

Issue the BBI command **.D A** to verify QAO is active.

3. Verify that the Rule Set and Rules are enabled.

Use the Rules Processor Automation Control panels to verify that the correct Rule Set and Rules are enabled and to alter the selection criteria or actions taken by a given Rule.

If the selection criteria for the Rule are too specific, the Rule may not fire for all the occasions that it is expected to. You can determine if this is the problem by relaxing the selection criteria in the control panels.

For more information, refer to the question “I have written a MQS Rule for an event that is not firing. What can I do?” on page A-14.

4. Verify that the queue manager is connected.

Examine the MAINVIEW AutoOPERATOR for MQSeries Workstation panel to determine if the queue manager is connected and verify the status of the instrumentation events.

For more information, refer to Question 2 on page A-8.

5. Determine which queues are eligible for automation.

schedule the QMQDSP00 EXEC to determine which queues are eligible for automation. This EXEC lists all the queues defined in a specific queue manager and which are currently enabled for automation.

For more information, refer to the question “How can I find out why MAINVIEW AutoOPERATOR is unable to monitor a particular queue?” on page A-11.

6. Verify that the OS/390 queue manager can be monitored.

Use MAINVIEW for WebSphere MQ to verify that the OS/390 queue manager is usable and has the correct attributes for automation.

7. Verify that the non-OS/390 queue manager can be monitored.

Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ to validate that the non-OS/390 queue manager is usable.

Refer to Figure A-1 on page A-3 to see how information flows to Rules and how commands flow to queue managers. Refer to “Step 3. Defining Connectivity” on page 2-14 to see how you must set up the paths for these flows to take place.

Frequently Asked Questions

Below is a list of questions frequently asked when verifying the installation and performance of MAINVIEW AutoOPERATOR for MQSeries. The diagnostic checklist shown on page A-4 is applied to these problems.

Question

After running the IVP EXEC QMQIVP00, how can I find out what is wrong with my installation of MAINVIEW AutoOPERATOR for MQSeries?

Answer

- A. Review the BBI-SS PAS' Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE AO for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the MAINVIEW AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- B. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- C. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed here:

```
QA1114E MQSERIES STEPLIB MISSING/INVALID
REASON  AO for MQ tried to initialize, but failed because the
STEPLIB did not specify a valid MQSeries load library.
SYSTEM  AO for MQ will not initialize.
ACTION
USER     Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION   statement contains the MQSeries Authlib.
```

- Step 1** If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review “Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 2-8.
- Step 2** Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 2-6.

If you have already followed the instructions, perform the following steps to ensure that the product was installed properly:

- 2.A** In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6.2.0 STARTED ON 28-JAN-2001 AT 09:53:11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, MAINVIEW AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries”.

- 2.B** In the BBI Journal, issue the BBI command **.D KEYS** (display keys).

This command shows all currently valid product option keys; for example:

```
CF8701I  COPY PROTECTION KEY STATUS:
CF8702I  PRD      CPUID      EXP      SYS      STATUS
CF8703I  -----  -----  -----  -----  -----
CF8704I  MAO?9672-15-*1317-99365-C4B0      VALID
CF8704I  MAO?9672-08-*3481-99365-4CA4      CPU-ERR
CF8704I  QAO-9672-15-11317-99365-D0A5      VALID
CF8704I  QAO-9672-08-13481-99365-D2DF      CPU-ERR
```

- 2.C** If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 2-6, or you do not have the key, contact BMC Software Customer Support to obtain the key.

- Step 3** Verifying that the Rule Set and Rules are enabled is not relevant in this situation.

Step 4 Verify the connection to the queue manager by using the MAINVIEW AutoOPERATOR for MQSeries Workstation panel and issue the BBI command `.RESET MQ xx`, where `xx` represents the two-character suffix of the `AAOMQLxx` member, to connect to the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to Chapter 4, “Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics”.

Step 5 You do not need to schedule the `QMQRSP00 EXEC` to determine which queues are available, because this information is not relevant.

Step 6 Use MAINVIEW for WebSphere MQ or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.

Step 7 Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

Question

How can I find out why the queue manager is not connected?

Answer

- A. Review the BBI-SS PAS’ Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE AO for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the MAINVIEW AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- B. Review the joblog messages for any error messages involving the queue manager and its channel initiator.

- C. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed here:

```
QA1114E  MQSERIES STEPLIB MISSING/INVALID
REASON   AO for MQ tried to initialize, but failed because the
STEPLIB did not specify a valid MQSeries load library.
SYSTEM   AO for MQ will not initialize.
ACTION
USER      Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION    statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review the “Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 2-8.

- Step 1** Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 2-6.

If you have already followed the instructions, perform the following to ensure that the product was installed properly:

- 1.A** In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6.2.0 STARTED ON 17-JAN-2001 AT 09:53:11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, MAINVIEW AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2.

1.B In the BBI Journal, issue the BBI command **.D KEYS** (display keys).

This command shows all currently valid product option keys; for example:

CF8701I	COPY PROTECTION KEY STATUS:				
CF8702I	PRD	CPUID	EXP	SYS	STATUS
CF8703I	-----	-----	-----	-----	-----
CF8704I	MAO?9672-15-*1317-99365-C4B0				VALID
CF8704I	MAO?9672-08-*3481-99365-4CA4				CPU-ERR
CF8704I	QAO-9672-15-11317-99365-D0A5				VALID
CF8704I	QAO-9672-08-13481-99365-D2DF				CPU-ERR

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 2-6, or you do not have the key, contact BMC Software Customer Support to obtain the key.

Step 2 Verifying that the Rule Set and Rules are enabled is not relevant in this situation.

Step 3 Verify the connection to the queue manager by using the MAINVIEW AutoOPERATOR for MQSeries Workstation panel and issue the BBI command **.RESET MQ xx**, where **xx** represents the two-character suffix of the **AAOMQLxx** member, to connect to the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to Chapter 4, “Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics”.

Step 4 You do not need to schedule the **QMJDSP00 EXEC** to determine which queues are available, because this information is not relevant.

Step 5 Use MAINVIEW for WebSphere MQ or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.

Step 6 Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

Question

How can I find out why MAINVIEW AutoOPERATOR is unable to monitor a particular queue?

Answer

- A. Review the BBI-SS PAS' Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE AO for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the MAINVIEW AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- B. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- C. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed here:

```
QA1114E  MQSERIES STEPLIB MISSING/INVALID
REASON  AO for MQ tried to initialize, but failed because the
STEPLIB did not specify a valid MQSeries load library.
SYSTEM  AO for MQ will not initialize.
ACTION
USER     Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION   statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review the "Specifying Product Option Password Keys" on page 2-6.

Step 1 Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 2-6.

If you have already followed the instructions, perform the following steps to ensure that the product was installed properly:

1.A In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6.2.0 STARTED ON 17-JAN-2001 AT 09:53:11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, MAINVIEW AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2.

1.B In the BBI Journal issue the BBI command **.D KEYS** (display keys).

This command shows all currently valid product option keys; for example:

```
CF8701I  COPY PROTECTION KEY STATUS:
CF8702I  PRD      CPUID      EXP      SYS      STATUS
CF8703I  -----
CF8704I  MAO?9672-15-*1317-99365-C4B0      VALID
CF8704I  MAO?9672-08-*3481-99365-4CA4      CPU-ERR
CF8704I  QAO-9672-15-11317-99365-D0A5      VALID
CF8704I  QAO-9672-08-13481-99365-D2DF      CPU-ERR
```

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 2-6, or you do not have the key, contact BMC Software Customer Support to obtain the key.

Step 2 Verify that a Rule exists that refers to the queue and the queue manager and that the Rule and its Rule Set are enabled.

Step 3 Verify the connection to the queue manager by using the MAINVIEW AutoOPERATOR for MQSeries Workstation panel and issue the BBI command **.RESET MQ xx**, where *xx* represents the two-character suffix of the AAOMQL*xx* member, to reset the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to “Chapter 4, “Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics”.

- Step 4** In the BBI Journal, issue the %QMCDSP00 command for a list of all queues and their status to determine which queues are eligible for automation. Find the specific queue in the list to verify that it will be monitored. An example of the output from the %QMCDSP00 EXEC follows:

```
.QMCDSP00 EID=00064 excl_by_user 32 local CSAE EPESIN.XMITQ  
.QMCDSP00 EID=00064 excl_by_user 32 remote CSAE CICSPRD1  
.QMCDSP00 EID=00064 excl_by_user 32 alias CSAE CICSPRD1.ALIASQ0  
.QMCDSP00 EID=00064 included 27 local CSAE CICSPRD1.QUEUE1  
.QMCDSP00 EID=00064 included 27 local CSAE CICSPRD1.QUEUE10  
.QMCDSP00 EID=00064 included 27 local CSAE CICSPRD1.QUEUE11
```

The queues named CICSPRD1.QUEUE* are eligible for automation.

If the queue is currently excluded, edit BBPARM member AAOMQLxx to add it to be included and issue a .RESET MQ xx command. Refer to “Parameters for BBPARM Member AAOMQLxx” on page 3-5 for more information.

- Step 5** Use MAINVIEW for WebSphere MQ or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.

Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ-Administrator to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

Question

I have written a MQS Rule for an event that is not firing. What can I do?

Answer

- A. Review the BBI-SS PAS' Start messages in the BBI Journal (or in the joblog) and look for the QA1113E message:

```
QA1113E UNABLE TO INITIALIZE AO for MQ TASK
```

The QA1113E error message occurs when you bring up a queue manager after the BBI-SS PAS is started and the system is unable to initialize the MAINVIEW AutoOPERATOR for MQSeries task. If you receive this error, you may need to contact BMC Software Customer Support for assistance.

If this message is not in the BBI Journal or the joblog, the BBI-SS PAS was started correctly.

- B. Review the joblog messages for any error messages involving the queue manager and its channel initiator.
- C. Verify that the MQSeries APF-authorized libraries are added to the STEPLIB DD CARD properly. If they are not added correctly or are missing, you may receive error message QA1114E. The error message and reason are displayed here:

```
QA1114E MQSERIES STEPLIB MISSING/INVALID
REASON  AO for MQ tried to initialize, but failed because the
STEPLIB did not specify a valid MQSeries load library.
SYSTEM  AO for MQ will not initialize.
ACTION
USER     Check the JCL for the BBI/SS to make sure the STEPLIB DD
ACTION   statement contains the MQSeries Authlib.
```

If the MQSeries libraries are missing from the STEPLIB DD concatenation, add them. The MQSeries libraries must be at or above the level of the OS/390 queue managers that are going to be automated. Also review the “Adding MQSeries APF-Authorized Libraries to the STEPLIB DD Card” on page 2-8.

- Step 1** Verify that the QAO product option key is properly installed and active. For information about how the QAO key is installed, refer to the section “Specifying Product Option Password Keys” on page 2-6.

If you have already followed the instructions, perform the following to ensure that the product was installed properly:

1.A In the BBI Journal, issue the BBI command **.D A** (display active).

This command shows all currently active products in the BBI-SS PAS; for example:

```
AA0101I  AAO VERSION 6.2.0 STARTED ON 17-JAN-2001 AT 09:53:11
AA0102I      CAO - ACTIVE
AA0103I      IAO - ACTIVE
AA0104I      MAO - ACTIVE
AA0106I      QAO - ACTIVE
```

If QAO is not listed as active, MAINVIEW AutoOPERATOR for MQSeries either has not been installed or has been installed incorrectly. For information about installing the product, refer to Chapter 2.

1.B In the BBI Journal, issue the BBI command **.D KEYS** (display keys).

This command shows all currently valid product option keys; for example:

```
CF8701I  COPY PROTECTION KEY STATUS:
CF8702I  PRD      CPUID      EXP      SYS      STATUS
CF8703I  -----
CF8704I  MAO?9672-15-*1317-99365-C4B0      VALID
CF8704I  MAO?9672-08-*3481-99365-4CA4      CPU-ERR
CF8704I  QAO-9672-15-11317-99365-D0A5      VALID
CF8704I  QAO-9672-08-13481-99365-D2DF      CPU-ERR
```

If you have typed an invalid key, as described in “Specifying Product Option Password Keys” on page 2-6, or you do not have the key, contact BMC Software Customer Support to obtain the key.

Step 2 Verify that the Rule Set and Rules are enabled.

Use the Rules Processor Automation Control panels to verify that the correct Rule Set and Rules are enabled and to alter the selection criteria or actions taken by a given Rule.

If the selection criteria for the Rule are too specific, the Rule may not fire for all the occasions that it is expected to. You can determine if this is the problem by relaxing the selection criteria in the control panels.

Step 3 Verify the connection to the queue manager by using the MAINVIEW AutoOPERATOR for MQSeries Workstation panel and issue the BBI command **.RESET MQ xx**, where **xx** represents the two-character suffix of the **AAOMQLxx** member, to reset the queue manager.

If the queue manager is not connected, follow the instructions in Chapter 2 to connect it.

For information about how to use the Workstation display, refer to Chapter 4, “Viewing MAINVIEW AutoOPERATOR for MQSeries Automation Statistics”.

- Step 4** In the BBI Journal, issue the %QMCDSP00 command for a list of all queues and their status to determine which queues are eligible for automation. Find the specific queue in the list to verify that it will be monitored. An example of the output from the %QMCDSP00 EXEC follows:

```
.QMCDSP00 EID=00064 excl_by_user 32 local CSAE EPESIN.XMITQ
.QMCDSP00 EID=00064 excl_by_user 32 remote CSAE CICSPRD1
.QMCDSP00 EID=00064 excl_by_user 32 alias CSAE CICSPRD1.ALIASQ0
.QMCDSP00 EID=00064 included 27 local CSAE CICSPRD1.QUEUE1
.QMCDSP00 EID=00064 included 27 local CSAE CICSPRD1.QUEUE10
.QMCDSP00 EID=00064 included 27 local CSAE CICSPRD1.QUEUE11
```

The queues named CICSPRD1.* are eligible for automation.

- Step 5** If the queue is currently excluded, edit BBPARM member AAOMQLxx to add it to be included. Refer to “Parameters for BBPARM Member AAOMQLxx” on page 3-5 for more information.

- Step 6** You can also use the %QMCDSP00 EXEC with a specific queue name:

```
%QMCDSP00 Q=BBOMVAO.YXP.QUEUE5
```

An example of the output from this EXEC follows:

```
.QMCDSP00 EID=00002 STARTED, PTF=BPO5319
.QMCDSP00 EID=00002 qmgr=CSBD is down
.QMCDSP00 EID=00002 qmgr=CSBE is down
.QMCDSP00 EID=00002 qmgr=CSBC is down
.QMCDSP00 EID=00002 excl_by_user 77 local CSQ1 BBOMVAO.YXP.QUEUE5
.QMCDSP00 EID=00002 excl_by_user 77 local CSQA BBOMVAO.YXP.QUEUE5
.QMCDSP00 EID=00002 qmgr=CSQ2 is down
.QMCDSP00 EID=00002 ENDED
```

- Step 7** Use MAINVIEW for WebSphere MQ or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.

- Step 8** Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

Question

I have written an EXEC where an IMFEXEC MQI PUT returns to me a non-zero condition code. What can I do?

Answer

Step 1 Look up any messages or codes.

Examine the BBI Journal and the ALERT Detail application for error messages or other indications that MAINVIEW AutoOPERATOR is not working correctly. For example, if an expected action did not occur, the BBI Journal may display whether or not the Rule fired that would have caused the action.

Refer to the *IBM MQSeries for OS/390 Messages and Codes* for more information about specific OS/390 messages. Depending on the platform, refer to the appropriate messages and codes book for non-OS/390 messages.

Step 2 If there are no messages, you can include the IMFEXEC DISPLAY MQ statement in the EXEC. This statement displays in the BBI Journal the values for all current variables. Verify that the variable's values are correct and appropriate.

Step 3 Verify that QAO is active since the MQCONN call worked.

Step 4 Do not verify that the Rule Set and Rules are enabled because this information is not relevant in this situation.

Step 5 Verify that the queue manager is connected.

Step 6 Do not schedule the QMQDSP00 EXEC to verify which queues are available, because this information is not relevant.

Step 7 Use MAINVIEW for WebSphere MQ or OS/390 commands to verify that the OS/390 queue manager is usable. Refer to the *IBM MQSeries for OS/390 Problem Determination Guide* for more information about any problems you may encounter.

Step 8 Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ to verify that the event queues are remote and working and the channels, transmission queue, and queue manager alias are working.

Question

What else can I do?

Answer

If you have completed the checklist and need additional information, perform the following actions:

Step 1 Use MAINVIEW for WebSphere MQ to obtain a general overview of the queues, channels, and queue managers being monitored by MAINVIEW AutoOPERATOR. The MAINVIEW for WebSphere MQ interface provides easy navigation to individual queues and queue managers and the ability to issue commands. You may also use the PATROL for WebSphere MQ interface for non-OS/390 queue managers.

Step 2 Hyperlink to the appropriate MAINVIEW for WebSphere MQ views to display the MQSeries objects and attributes and overwrite their values to update them.

MAINVIEW for WebSphere MQ issues the appropriate commands depending on whether you are listening to OS/390 or non-OS/390 queue managers. For a complete list of commands, see *MQSeries Command Reference* and the *MQSeries System Management Guide* for the appropriate platform.

Step 3 Display local queue manager's objects to determine if they are correctly defined and active.

Refer to the *MQSeries for OS/390 Problem Determination Guide*.

- Use MAINVIEW for WebSphere MQ to inspect queues and channels.
- Use SDSF (System Display and Search Facility) or a similar application to issue MQSeries commands to verify that the queue manager and its channel initiator are running correctly.

Step 4 Display remote queue manager's objects to determine if they are correctly defined and active.

- Use MAINVIEW for WebSphere MQ or PATROL for WebSphere MQ to inspect queues and channels.
- Refer to the platform-specific system management guides.
- Use platform-specific commands to verify that the queue manager is active and its channel initiator is running.

Examples of Specific Automation Problems and Resolutions

This section provides examples of more specific automation problems and the steps taken to resolve them.

Dead Letter Queue Solution

My DLQ Solution Rule does not move normal messages from the dead letter queue. What do I do?

Follow the “Diagnostic Checklist” on page A-4 as follows:

Step 1 Determine if you have received any messages or codes. For this scenario, there are no messages or codes to look up.

Step 2 Issue the **.D A** command to verify that MAINVIEW AutoOPERATOR for MQSeries is installed and that the QAO product option key is implemented properly.

Again, for this scenario, the product and the key are properly installed.

Step 3 Use the Automation Control panel to inspect the Rule Set and Rules to verify that the Rule Set and the DLQ Rule are both enabled. Also check the automation strategy setting.

The Automation Control panel shows that the Rule Set is enabled and the automation strategy is set to ALL.

If the automation strategy was set to FIRST, you might need to review the individual Rules and ensure that DLQ Rule is high enough in the Rule Set for a FIRST strategy to work.

If the automation strategy was set to MOST QUALIFIED, you should also review the individual Rules and ensure that there is not another Rule in the Rule Set that has more matching selection criteria set.

Figure A-2 Automation Control Panel

```

BMC Software ----- Automation Control ----- AutoOPERATOR
COMMAND ==>                                     TGT ==> SYSE
Primary commands: Add, Statshow, Cmdshow         DATE --- 01/02/22
                                                TIME --- 11:19:08

Automation Status ==> ACTIVE                    (Active, Inactive)
Automation Strategy ==> ALL                      (Individual, All, First, Qualified)
Honor MPF Suppression ==> YES                   (NO/YES)

Automation Library
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (SA)ve
           (M)ove, (B)efore or (A)fter, (F)ilter Criteria
LC   Rule-Set Status  Rules    Fired  Filtered  Date      Time      Strategy
___  AAORULBQ ENABLED   16       0       0  22-FEB-01 11:17:54 ALL
___  AAORULBA DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBB DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBC DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBD DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBE DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBF DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBG DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBH DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBP DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBR DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBS DISABLED  N/A     N/A     N/A      N/A      N/A
___  AAORULBV DISABLED  N/A     N/A     N/A      N/A      N/A
    
```

Step 4 Select the Rule Set.

Figure A-3 on page A-21 shows that the three DLQ Rules MQDEDQ01, MQDEDQ02, and MQDEDQ03 are enabled.

Figure A-3 Rule Set Overview Panel

```

BMC Software ----- Rule Set Overview -----
AutoOPERATOR
COMMAND ==> TGT --- SYSE
Rule Set ID: AAORULBQ DATE ---
01/02/22
Primary commands: Add, Save, Sort, Unsort, Reset, Filter TIME ---
11:24:10
LC CMDS --- (S)elect, (E)nable, (D)isable, (T)est, (DE)lete, (I)nsert
(C)opy/(CC)opy, (M)ove/(MM)ove, (B)efore or (A)fter, (R)peat
Sort Criterion: Filter ENABLED right/left

LC Rule-id Stat Text-id Type Fired EXEC Changed ID
-----
___ MQINT001 DIS Q_SERVICE_INTERV MQS 0 97/03/14 11:09 BMC1
___ MQINT002 DIS Q_SERVICE_INTERV MQS 0 97/03/14 11:10 BMC1
___ MQQDP001 DIS Q_DEPTH_HIGH MQS 0 97/03/14 11:10 BMC1
___ MQQDP002 DIS Q_DEPTH_LOW MQS 0 97/03/14 11:10 BMC1
___ MQDEDQ01 ENA MQS 0 01/02/14 12:34
BMCUSER
___ MQDEDQ02 ENA TIME 0 QMQDEDQ2 01/02/14 07:53
BMCUSER
___ MQDEDQ03 ENA MQS 0 01/02/14 12:35
BMCUSER
___ MQARC001 ENA MQS 0 QMQUNLDQ 01/02/12 13:20
BMCUSER
___ MQARC002 ENA QA6033I JRNL 0 01/02/12 15:09
BMCUSER
___ MQARC003 ENA QA6033I JRNL 0 01/02/12 15:19
BMCUSER
___ MQARC004 ENA QA6033I JRNL 0 01/02/12 13:24
BMCUSER
___ MQARC005 ENA QA6033I JRNL 0 01/02/16 09:27
BMCUSER
___ MQDIA001 ENA UNKNOWN_OBJECT_N MQS 0 97/03/07 09:30 BMC1
___ MQDIA002 ENA NOT_AUTHORIZED MQS 0 97/03/14 09:24 BMC1
    
```

Editing Rule MQDEDQ01 shows the Rule has the correct queue manager and queue name specified.

Figure A-4 Selection Criteria Panel

```

BMC Software ----- Selection Criteria - MQS ----- AutoOPERATOR
COMMAND ==>                                           TGT --- SYSE

                Rule-set === AAORULBQ                Rule-id === MQDEDQ01

Queue Identification:                                (1 to 12 Queue Managers)
Manager(s)    ==> CSBD
Queue Id      ==> &!QDED.IMFOQMGR

Message Identification:
Format        ==> USER                               (Value from MD Format field)
Event Type    ==>                                     (Enter ? for help)

                Sub  Len  Op  Value
Msgid         ==> ___ : ___ - _____
CorrelId      ==> ___ : ___ - _____
Msg Buffer     ==> ___ : ___ - _____

Press ENTER to continue, END return to Detail Control, CANCEL to cancel changes
    
```

See Chapter 4 for descriptions of the shared variable QDEDxxxx (the name of the dead letter queue) and the TSO variable, IMFQRPQM (the name of the queue manager).

Step 5 Use the MAINVIEW AutoOPERATOR for MQSeries Workstation panel to determine if the queue manager is connected and enabled for automation.

This panel shows that MAINVIEW AutoOPERATOR is connected to the queue manager (CSQ1) and is monitoring 15 queues:

```

BMC Software ----- MQSeries Workstation ----- AutoOPERATOR
COMMAND ==>                                           TGT ==> SYSE
Interval ==> 3      INPUT                               DATE --- 01/02/22
Commands: ALL, EQI xxx, NEQI xxx, EE xxx, NEE xxx      TIME --- 16:00:28
----- Performance Statistics -----
Total Queues                228  Total Queues Included                37
Total MQS Messages Included  31  Total MQS Messages Handled           31
Instrument. Events Included  31  Instrument. Events Handled           31
User Messages Included       0  User Messages Handled                 0
Current Instr.Arrival Rate/Min 0  Peak Instr. Arrival Rate/Min         27
Rule Generated Alerts        31  Rule Triggered EXECs                 31
----- Queue Management -----
Queue Manager                Queues   Event Ques  Enabled
                             Incl./Total Included  Events
-----
CSQ1                          37/228    QCP        SLIPR
***** Bottom of Data *****
    
```

Step 6 Issue the BBI command **.D V QDED.CSQ1** to display the name of the dead letter queue for your OS/390 queue manager. The output will look like this panel:

```

BMC Software ----- Log Display ----- General services
COMMAND ===>                                     TGT ===> LOCAL
Line      2 Log #1 Status INPUT      Time 16:07:06 INTV===> 3
16:07:05 .D V QDED.CSQ1
16:07:06 IM9100I  COMMAND ACCEPTED                JB61
16:07:06 IM9207I  SHARED VARIABLE POOL DISPLAY    JB61
16:07:06 IM9211I  SELECTED: QDED.CSQ1                    JB61
16:07:06 IM9212I  QDED.CSQ1                        DEAD.LETTER.QUEUE JB61
***** END OF LOG *****

```

Step 7 Type the **%QMQDSP00** command in the BBI Journal to check the status of the dead letter queue.

The dead letter queue (DEAD.LETTER.QUEUE) is not eligible for automation.

```
.QMQDSP00 EID=00021 excl_by_user 17 local CSQ1 DEAD.LETTER.QUEUE
```

You can also edit and review BBPARM member AAOMQL00 to determine if the dead letter queue has been included. As you can see from the picture below, an incorrect dead letter queue name was used and needs to be changed to DEAD.LETTER.QUEUE.

```

TYPE(INCL) QMGR(CS*) QUEUE(SYSTEM.ADMIN.*)
TYPE(INCL) QMGR(CS*) QUEUE(SYSBMC.SYSE.EVENTS)
TYPE(INCL) QMGR(CS*) QUEUE(SYSBMC.DISTRIB.EVENTS)
TYPE(INCL) QMGR(CS*) QUEUE(BBOMVAO.QUEUE*)
TYPE(EXCL) QMGR(CS*) QUEUE(BBOMVAO.SETUP.QUEUE*)
TYPE(INCL) QMGR(CS*) QUEUE(BBOMVAO.LIVE.*)
TYPE(INCL) QMGR(CS*) QUEUE(BBOMVAO.PERM.QUEUE)
TYPE(INCL) QMGR(CS*) QUEUE(SYSTEM.DEAD.LETTER.QUEUE)
T(E) QMGR(*) QUEUE(*)

```

Note: You must issue the BBI control command **.RESET MQ 00** to activate the AAOMQL00 member and at least one enabled Rule must refer to this queue manager for MAINVIEW AutoOPERATOR to connect to the queue manager.

The rule did not move this user message to another queue because the wrong name was used in the AAOMQLnn BBPARM member to refer to the active dead letter queue.

Step 8 Change the AAOMQL00 BBIPARM member to include SYSTEM.DEAD.LETTER.QUEUE instead of DEAD.LETTER.QUEUE.

Step 9 Issue the `.E MQ 00` command.

The Rule now fires.

Non-OS/390 Command Times Out

My EXEC times out when it issues a display command to a non-OS/390 queue manager. How do I fix this?

Use the “Diagnostic Checklist” on page A-4 as a guideline when completing these steps:

Step 1 Look up the code X'7F1', which is displayed by the EXEC in the BBI Journal.

```
%MQRNTCMD
EM9201I MQGET 0001 LEN=X'0000' CC=X'0002' RC=X'000007F1'
EM0022E ERROR PROCESSING .. MQRNTCMD .. TIMED OUT WAITING FOR RSP
EM0025I FOLLOWING MSG ISSUED FOR EXEC .. MQRNTCMD ..
      11 *-* "INFEXEC CMD 'DISPLAY QMGR ALL' TYPE (MQS) QM("mvs_qm") PC
      LM("mvs_qm") CQ (SYSTEM.ADMIN.COMMAND.QUEUE) "
      +++ RC (4) +++
```

According to *MQSeries for OS/390 Messages and Codes*, X'7F1' means a MQGET request timed out.

Step 2 We know the QAO is active because the EXEC used the MQI successfully.

Step 3 Since there are no Rules to consider, you do not need to refer to the Rules Processor Automation Control panels.

Step 4 Look at the MAINVIEW AutoOPERATOR for MQSeries OCD.

Step 5 The panel indicates that MAINVIEW AutoOPERATOR is connected to queue manager CSQ4, which is correct.

Step 6 There are no Rules, so you do not need to determine which queues are eligible for automation.

Step 7 Return to MAINVIEW for WebSphere MQ to complete the following procedures:

Step 8 Validate the OS/390 queue manager.**8.A** Obtain the name of the transmission queue.

In this case, the queue manager alias is PCOCHRAN and its transmission queue is CSQ4.

```

25FEB1997 14:55:53 ----- INFORMATION DISPLAY -----
COMMAND ==>
CURR WIN ==> 1          ALT WIN ==>
W1 =RQD=====CSQ4====CSQ4====-25FEB1997==14:55:52====MUMQS=====1=
Queue..... PCOCHRAN
Description..... (none)
Queue Manager Name.. CSQ4
Transmission Queue.. PCOCHRAN.XMITQ

```

8.B Verify that the XMITQ is empty by hyperlinking from XMITQ to a detailed view.

```

25FEB1997 14:59:12 ----- INFORMATION DISPLAY -----
COMMAND ==>
CURR WIN ==> 1          ALT WIN ==>
W1 =LQD=====CSQ4====CSQ4====-25FEB1997==14:59:09====MUMQS=====1=
Current Depth..... 0      Queue..... PCOCHRAN.XMITQ
Maximum Depth..... 99999999 Description..... (none)

```

8.C Verify that the Sender channel is running by hyperlinking to the CHNLS view.

```

25FEB1997 14:50:08 ----- INFORMATION DISPLAY -----
COMMAND ==>
CURR WIN ==> 1          ALT WIN ==>
W1 =CHNLS====CHNLAD====CSQ4====CSQ4====-25FEB1997==14:49:00====MUMQS=====1=
Channel Name..... CSQ4.TO.PCOCHRAN Description... (none)
Type..... SENDER      Queue Manager.. CSQ4
Status..... RUNNING   Xmit Queue.... PCOCHRAN.XMITQ
Put Authority.. CONTEXT

```

8.D Verify that the Receiver Channel is running.

```

25FEB1997 16:16:44 ----- INFORMATION DISPLAY -----
COMMAND ==>
CURR WIN ==> 1          ALT WIN ==>
W1 =CHNLS====CHNLAD====CSQ4====CSQ4====-25FEB1997==16:13:41====MUMQS=====1=
Channel Name..... PCOCHRAN.TO.CSQ4 Description... RECEIVER CHANNEL FOR PCOCHRAN (NT)
Type..... RECEIVER    Queue Manager.. CSQ4
Status..... RUNNING   Xmit Queue.... N/A

```

Step 9 Validate the non-OS/390 queue manager.

In this example, we are using native MQSeries commands on Windows NT.

9.A Verify that the command queue is empty and enabled.

```
dis q(system.admin.command.queue) curdepth put get
3 : dis q(system.admin.command.queue) curdepth put get
AMQ8409: Display Queue details.
  QUEUE(SYSTEM.ADMIN.COMMAND.QUEUE)
  GET(ENABLED)
  PUT(ENABLED)
  CURDEPTH(0)
```

9.B Stop the command server and reissue the command.

The command is present in the queue.

```
dis q(system.admin.command.queue) curdepth
1 : dis q(system.admin.command.queue) curdepth
AMQ8409: Display Queue details.
  QUEUE(SYSTEM.ADMIN.COMMAND.QUEUE)
  CURDEPTH(1)
```

9.C Restart the command server.**9.D** Determine if the queue manager alias is PUT inhibited.

```
dis q(csq4) put
16 : dis q(csq4) put
AMQ8409: Display Queue details.
  QUEUE(CSQ4)
  PUT(DISABLED)
```

9.E Alter the queue manager alias to PUT (ENABLED).

The non-OS/390 command no longer times out.

Rule Will Not Enable a Queue

A Rule will not enable a non-OS/390 queue. What is the solution?

- Step 1** Write the text of the command issued by the Rule as an ALERT to verify that the command syntax is correct.
- Step 2** Use MAINVIEW for WebSphere MQ to verify that the channel to the non-OS/390 queue manager is running.

```
W1 =CHNLS====CHNLAD===CSQ4====*=====25FEB1997--23:16:30===MUMQS=====1=
Channel Name..... CSQ4.TO.PCOCHRAN      Description... (none)
Type..... SENDER                          Queue Manager.. CSQ4
Status..... RUNNING                       Xmit Queue.... PCOCHRAN.XMITQ
```

- Step 3** Verify that the transmission queue is empty and enabled by hyperlinking from its name to a detailed view of its attributes.

```
W1 =LQD=====CSQ4====CSQ4====25FEB1997--23:13:13===MUMQS=====1==
Current Depth..... 0                      Queue..... PCOCHRAN.XMITQ
Maximum Depth..... 99999999              Description..... (none)
Inhibited Actions....                    Trigger.....
Gets..... No                             Control..... On
Puts..... No                             Type..... First
```

- Step 4** Determine the status of the non-OS/390 command queue. Use the MQSC display commands for the queue manager on the distributed platform.

```
dis q(system.admin.command.queue) curdepth put get
1 : dis q(system.admin.command.queue) curdepth put get
AMQ8409: Display Queue details.
QUEUE(SYSTEM.ADMIN.COMMAND.QUEUE)
GET(ENABLED)
PUT(ENABLED)
CURDEPTH(126)
```

- Step 5** Verify that the remote command queue is GET and PUT enabled, but CURDEPTH is not zero.
- Step 6** Start the command server on the non-OS/390 queue manager.

The command from the Rule will now execute successfully for the non-OS/390 queue.

Appendix B MQI Sample Coding

To help users write MQ Automation more quickly, MAINVIEW AutoOPERATOR for MQSeries provides sample code for the more common MAINVIEW AutoOPERATOR commands. These examples contain a shell consisting of all the pieces necessary to execute a MQSeries request, such as putting a message to a queue or manipulating a MQSeries resource. The example code can be copied or cut and pasted into your EXECs, where you can then modify the examples, or portions of the examples, to fit the needs for your particular request.

These examples are located in BBSAMP and are named QMQNB001 – QMQNB009.

Table B-1 lists the nine examples and the actions that they perform.

Table B-1 MQI Code Examples (Part 1 of 2)

Example Name	Description
QMQNB001	puts a message to a queue on an OS/390 queue manager The code is divided into five sections that each complete a specific task: <ul style="list-style-type: none">• Connects to a queue manager• Opens a MQSeries queue• PUTs a message to the queue• Closes the queue• Disconnects from the queue manager
QMQNB002	puts a message to a queue on an OS/390 queue manager using put1 The code is divided into three tasks: <ul style="list-style-type: none">• Connects to a queue manager• PUT1s a message to a queue• Disconnects from the queue manager

Table B-1

MQI Code Examples (Part 2 of 2)

Example Name	Description
QMQNB003	Browses a local queue defined to an OS/390 queue manager The code is divided into five tasks: <ul style="list-style-type: none"> • Connects to a queue manager • Opens a MQSeries queue • Gets a message for browse from an opened queue • Closes the queue • Disconnects from the queue manager
QMQNB004	puts a message to two different queues using syncpoint The code is divided into nine tasks: <ul style="list-style-type: none"> • Connects to a queue manager • Opens the first queue • PUTS a message to the first queue and begins syncpoint processing • Closes that queue • Opens the second queue • PUTS a message to the second queue using the same syncpoint • Closes the second queue • Commits all work since last syncpoint • Disconnects from the queue manager
QMQNB005	Displays the status of a queue manager resource The code contains one task: Displays the status of performance events for the queue manager
QMQNB006	Alters the status of a queue manager The code is contains one task: Alters the QMGR to allow INHIBIT events for the queue manager
QMQNB007	Alters the status of a queue The code contains one task: Alters a queue to allow Q_Depth_High monitoring and to set a threshold of 80%
QMQNB008	GETS 10 messages for browse from a queue and writes the message text to the BBI Journal The code is divided into five tasks: <ul style="list-style-type: none"> • Connects to the queue manager • Opens a queue • GETS 10 messages from the queue and displays each of them in the BBI Journal • Closes the queue • Disconnects the queue manager
QMQNB009	Contains a diagnostic procedure that can be copied into any EXEC that uses MAINVIEW AutoOPERATOR MQI or MAINVIEW AutoOPERATOR for MQSeries command facility The code is divided into three tasks: <ul style="list-style-type: none"> • Displays non-zero return codes • Displays MQ messages • Returns to caller

In the following example, QMQNB001 PUTs a message to a queue on an OS/390 queue manager:

```
/*-----REXX-----*/
/* QMQNB001 - */
/* This sample PUTs a message to a queue on an OS/390 Queue Manager. */
/* Keyword parameters will be used to set the appropriate values for */
/* input to each call to the MQI. */
/* Change Activity: */
/* 09-29-2000: Updated for new interface */
/*-----*/

/*-----*/
/* This statement causes a connection to the Queue Manager. */
/* NAME = Queue Manager ID */
/*-----*/
"IMFEXEC MQI CONN NAME(mqid)"

/*-----*/
/* This statement opens a MQSeries Queue. */
/* HOBJ = Queue name */
/* HCONN = connection handle (can be omitted, default is IMFHCONN) */
/* OOPTS = queue is opened for output */
/*-----*/
IMFMQI_OD_OBJECTNAME = qname
"IMFEXEC MQI OPEN HCONN(IMFHCONN) OOPTS(MQOO_OUTPUT)"

/*-----*/
/* This statement PUTs a message to a previously opened queue. */
/* HCONN = connection handle (can be omitted, default is IMFHCONN) */
/* HOBJ = queue name, set by previous OPEN */
/* POPTS = PUT options */
/* BUFFLEN = length of variable containing application data */
/* BUFFER = name of variable that contains application data */
/*-----*/
"IMFEXEC MQI PUT HCONN(IMFHCONN) HOBJ(IMFHOBJ) POPTS(MQPMO_NO_SYNCPOINT)",
"BUFFLEN("LENGTH(message)") BUFFER(message)"

/*-----*/
/* This statement closes a MQSeries Queue. */
/* HOBJ = Queue name */
/* HCONN = connection handle (can be omitted, default is IMFHCONN) */
/* COPTS = None is specified for pre-defined queues */
/*-----*/
"IMFEXEC MQI CLOSE COPTS(MQCO_NONE) HOBJ(IMFHOBJ)"

/*-----*/
/* This statement causes a disconnection from the Queue Manager. */
/* HCONN = Connection handle set during previous connect */
/*-----*/
"IMFEXEC MQI DISC HCONN(IMFHCONN)"
```



Appendix C Rules Variables

EVENT Variables – Rules and Rule-Invoked EXECs

This table lists the event variables - Rules and Rule-Invoked EXECs:

Table C-1 EVENT Variables – Rules and Rule-Invoked EXECs (Part 1 of 2)

Variable Name	Notes
IMFQ_EVENT_TYPE	Event type-not a MQSeries field
IMFQ_EVENT_QMGRNAME	Name of queue manager that generated event, whether local or remote
IMFQ_EVENT_QNAME	Name of queue from object descriptor
IMFQ_EVENT_BASEQNAME	Name of queue manager (to which the alias resolves)
IMFQ_EVENT_APPLNAME	Name of application
IMFQ_EVENT_OBJECTQMGRNAME	Name of the object queue manager
IMFQ_EVENT_CHANNEL_NAME	Name of channel
IMFQ_EVENT_CONNECTIONNAME	Connection name. For TCP/IP, this is the internet address only if the channel has successfully established a connection. Otherwise, it is the contents of the ConnectionName field in the channel definition.
IMFQ_EVENT_XMITQNAME	Transmission queue name
IMFQ_EVENT_FORMAT	Name of format
IMFQ_EVENT_QTYPE	Type of queue
IMFQ_EVENT_APPLTYPE	Type of queue to which the alias resolves (A for alias, M for model)
IMFQ_EVENT_REASONQUALIFIER	Reason qualifier: MQRQ_CHANNEL_STOPPED_OK MQRQ_CHANNEL_STOPPED_ERROR MQRQ_CHANNEL_STOPPED_RETRY MQRQ_CHANNEL_STOPPED_DISABLED

Table C-1 **EVENT Variables – Rules and Rule-Invoked EXECs (Part 2 of 2)**

Variable Name	Notes
IMFQ_EVENT_ERRORIDENTIFIER	See the section regarding the CHANNEL_STOPPED event, field Error identifier, in the IBM publication <i>MQSeries Programmable System Management</i> for a description of this data.
IMFQ_EVENT_PROCESSNAME	Name of process
IMFQ_EVENT_USERIDENTIFIER	Name of user identifier
IMFQ_EVENT_AUXERRORDATAINT1	First integer of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATAINT2	Second integer of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATASTR1	First string of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATASTR2	Second string of auxiliary error data for channel errors
IMFQ_EVENT_AUXERRORDATASTR3	Third string of auxiliary error data for channel errors
IMFQ_EVENT_TIMESINCERESET	Time (in seconds) since the statistics were last reset; for this event, this value is greater than the service interval
IMFQ_EVENT_HIGHQDEPTH	Max number of messages on the queue since the statistics were last reset
IMFQ_EVENT_MSGENQCOUNT	Number of messages enqueued since statistics were last reset
IMFQ_EVENT_MSGDEQCOUNT	Number of messages dequeued since statistics were last reset
IMFQ_EVENT_BRIDGENAME	Name of bridge
IMFQ_EVENT_BRIDGETYPE	Only current value is 1 (OTMA)
IMFQ_EVENT_OPTIONS	Open options
IMFQ_EVENT_COMMAND	Name of command event
IMFQ_EVENT_CURDEPTH	Current depth of the event queue
IMFQ_EVENT_MAXDEPTH	Maximum depth of the event queue
IMFQ_EVENT_PERCENTFULL	Percent full of the event queue

Dead Letter Header Variables – Rules and Rule-Invoked EXECs

This table lists the Dead Letter Header variables – Rules and Rule-Invoked EXECs:

Table C-2 Dead Letter Header Variables – Rules and Rule-Invoked EXECs (Part 1 of 2)

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_DLH_STRUCID	Contains MQDLH_STRUC_ID.
IMFQ_DLH_VERSION	Contains MQDLH_VERSION_1.
IMFQ_DLH_REASON	Contains a numeric reason code between 0 and 999999999.
IMFQ_DLH_DESTQNAME	Contains the 48-character destination queue name.
IMFQ_DLH_DESTQMGRNAME	Contains the 48-character destination queue manager name.
IMFQ_DLH_ENCODING	Contains decimal data.
IMFQ_DLH_CODEDCHARSETID	Contains decimal data.
IMFQ_DLH_FORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2

Table C-2 Dead Letter Header Variables – Rules and Rule-Invoked EXECs (Part 2 of 2)

Variable Name	Notes
IMFQ_DLH_PUTAPPLTYPE	<p>Contains a numeric user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> MQAT_AIX MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DOS MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <p>See note at the beginning of this table.</p>
IMFQ_DLH_PUTAPPLNAME	Contains up to a 28-character name for PutAppName.
IMFQ_DLH_PUTDATE	Contains an 8-character date stamp.
IMFQ_DLH_PUTTIME	Contains an 8-character time stamp.

Trigger Message Variables – Rules and Rule-Invoked EXECs

This table lists the Trigger Message variables – Rules and Rule-Invoked EXECs:

Table C-3 Trigger Message Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_TM_STRUCID	Contains MQTM_STRUC_ID.
IMFQ_TM_VERSION	Contains MQTM_VERSION_1.
IMFQ_TM_QNAME	Contains 48-character name of triggered queue.
IMFQ_TM_PROCESSNAME	Contains 48-character name of process object.
IMFQ_TM_TRIGGERDATA	Contains 64-character free-format trigger data.
IMFQ_TM_APPLTYPE	<p>Contains a numeric user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> MQAT_AIX MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DOS MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <p>See note at the beginning of this table.</p>
IMFQ_TM_APPLID	Contains a 256-character string.
IMFQ_TM_ENVDATA	Contains a 128-character string.
IMFQ_TM_USERDATA	Contains a 128-character string.

CICS Information Header Variables – Rules and Rule-Invoked EXECs

This table lists the CICS Information Header variables – Rules and Rule-Invoked EXECs:

Table C-4 CICS Information Header Variables – Rules and Rule-Invoked EXECs (Part 1 of 4)

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_CIH_STRUCID	Contains the value MQCIH_STRUC_ID.
IMFQ_CIH_VERSION	Contains the value MQCIH_VERSION_2.
IMFQ_CIH_STRUCLength	Contains the value MQCIH_LENGTH_2.
IMFQ_CIH_ENCODING	Contains the decimal 0.
IMFQ_CIH_CODEDCHARSETID	Contains the decimal 0.
IMFQ_CIH_FORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_CIH_FLAGS	Contains MQCIH_NONE.

Table C-4 CICS Information Header Variables – Rules and Rule-Invoked EXECs (Part 2 of 4)

Variable Name	Notes
IMFQ_CIH_RETURNCODE	Contains one of the following values: MQCRC_OK MQCRC_CICS_EXEC_ERROR MQCRC_MQ_API_ERROR MQCRC_BRIDGE_ERROR MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_BRIDGE_TIMEOUT MQCRC_TRANSID_NOT_AVAILABLE
IMFQ_CIH_COMPCODE	Contains decimal data.
IMFQ_CIH_REASON	Contains decimal data.
IMFQ_CIH_UOWCONTROL	Contains one of the following values: MQCUOWC_ONLY MQCUOWC_CONTINUE MQCUOWC_FIRST MQCUOWC_MIDDLE MQCUOWC_LAST MQCUOWC_COMMIT MQCUOWC_BACKOUT
IMFQ_CIH_GETWAITINTERVAL	Contains a decimal or one of the following constants: MQCGWI_DEFAULT MQWI_UNLIMITED
IMFQ_CIH_LINKTYPE	Contains one of the following values: MQCLT_PROGRAM MQCLT_TRANSACTION
IMFQ_CIH_OUTPUTDATALENGTH	Contains decimal data or the constant: MQCLT_TRANSACTION
IMFQ_CIH_FACILITYKEEPTIME	Contains decimal data.
IMFQ_CIH_ADSD DESCRIPTOR	Contains one of the following values: MQCADSD_NONE MQCADSD_SEND MQCADSD_RECV MQCADSD_MSGFORMAT <i>See note at the beginning of this table.</i>
IMFQ_CIH_CONVERSATIONALTASK	Contains one of the following values: MQCCT_YES MQCCT_NO

Table C-4 CICS Information Header Variables – Rules and Rule-Invoked EXECs (Part 3 of 4)

Variable Name	Notes
IMFQ_CIH_TASKENDSTATUS	Contains one of the following values: MQCTES_NOSYNC MQCTES_COMMIT MQCTES_BACKOUT MQCTES_ENDTASK
IMFQ_CIH_FACILITY	Contains a 8-byte hexadecimal or the following constant: MQCFAC_NONE. No conversion of hexadecimal data is done.
IMFQ_CIH_FUNCTION	Contains a 4-character value of one of the following constants: MQCFUNC_MQCONN MQCFUNC_MQGET MQCFUNC_MQINQ MQCFUNC_MQOPEN MQCFUNC_MQPUT MQCFUNC_MQPUT1 MQCFUNC_NONE <i>See note at the beginning of this table.</i>
IMFQ_CIH_ABENDCODE	Contains a 4-character value.
IMFQ_CIH_AUTHENTICATOR	Contains an 8-character value.
IMFQ_CIH_RESERVED1	Contains 8 blanks.
IMFQ_CIH_REPLYTOFORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_CIH_REMOTESYSID	Contains 4 blanks.
IMFQ_CIH_REMOTETRANSID	Contains 4 blanks.

Table C-4 CICS Information Header Variables – Rules and Rule-Invoked EXECs (Part 4 of 4)

Variable Name	Notes
IMFQ_CIH_TRANSACTIONID	Contains a 4-character value.
IMFQ_CIH_FACILITYLIKE	Contains a 4-character value.
IMFQ_CIH_ATTENTIONID	Contains a 4-character value.
IMFQ_CIH_STARTCODE	<p>Contains one of the following constants:</p> <p>MQCSC_START MQCSC_STARTDATA MQCSC_TERMINPUT MQCSC_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_CIH_CANCELCODE	Contains a 4-character value.
IMFQ_CIH_NEXTTRANSACTIONID	Contains a 4-character value.
IMFQ_CIH_RESERVED2	Contains 8 blanks.
IMFQ_CIH_RESERVED3	Contains 8 blanks.
IMFQ_CIH_CURSORPOSITION	Contains decimal data.
IMFQ_CIH_ERROROFFSET	Contains decimal data.
IMFQ_CIH_INPUTITEM	Contains the decimal 0.
IMFQ_CIH_RESERVED4	Contains the decimal 0.

IMS Information Header Variables – Rules and Rule-Invoked EXECs

This table lists the IMS Information Header variables – Rules and Rule-Invoked EXECs:

Table C-5 IMS Information Header Variables – Rules and Rule-Invoked EXECs (Part 1 of 2)

Variable Name	Notes
IMFQ_IIH_STRUCID	Contains MQIIH_STRUC_ID.
IMFQ_IIH_VERSION	Contains MQIIH_VERSION_1.
IMFQ_IIH_STRUCLength	Contains MQIIH_LENGTH_1.
IMFQ_IIH_ENCODING	Contains the decimal 0.
IMFQ_IIH_CODEDCHARSETID	Contains the decimal 0.
IMFQ_IIH_FORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_IIH_FLAGS	Contains MQIIH_NONE.
IMFQ_IIH_LTERM_OVERRIDE	Contains an 8-character value.
IMFQ_IIH_MFSMAPNAME	Contains an 8-character value.

Table C-5 IMS Information Header Variables – Rules and Rule-Invoked EXECs (Part 2 of 2)

Variable Name	Notes
IMFQ_IIH_REPLYTOFORMAT	Contains one of the following format constants: <ul style="list-style-type: none"> MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_IIH_AUTHENTICATOR	Contains an 8-character value or the value MQIAUT_NONE.
IMFQ_IIH_TRANINSTANCEID	Contains a 16-byte hexadecimal or the constant MQITII_NONE. No conversion of hexadecimal data is done.
IMFQ_IIH_TRANSTATE	Contains one of the following constants: <ul style="list-style-type: none"> MQITS_IN_CONVERSATION MQITS_NOT_IN_CONVERSATION MQITS_ARCHITECTED
IMFQ_IIH_COMMITMODE	Contains one of the following constants: <ul style="list-style-type: none"> MQICM_COMMIT_THEN_SEND MQICM_SEND_THEN_COMMIT
IMFQ_IIH_SECURITYSCOPE	Contains one of the following constants: <ul style="list-style-type: none"> MQISS_CHECK MQISS_FULL
IMFQ_IIH_RESERVED	Contains one blank.

Transmission Queue Header Variables – Rules and Rule-Invoked EXECs

This table lists the Transmission Queue Header variables – Rules and Rule-Invoked EXECs:

Table C-6 Transmission Queue Header Variables – Rules and Rule-Invoked EXECs (Part 1 of 5)

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_XQH_STRUCID	Contains MQXQH_STRUC_ID.
IMFQ_XQH_VERSION	Contains MQXQH_VERSION_1.
IMFQ_XQH_REMOTEQNAME	Contains a 48-character value.
IMFQ_XQH_REMOTEQMGRNAME	Contains a 48-character value.
IMFQ_XQH_MD_STRUCID	Contains MQMD_STRUC_ID.
IMFQ_XQH_MD_VERSION	Contains MQMD_VERSION_1.
IMFQ_XQH_MD_REPORT	<p>Contains one or more of the following constants delimited by blanks:</p> <p>MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE</p> <p><i>See note at the beginning of this table.</i></p>

Table C-6 Transmission Queue Header Variables – Rules and Rule-Invoked EXECs (Part 2 of 5)

Variable Name	Notes
IMFQ_XQH_MD_MSGTYPE	Contains one of the following constants: MQMT_DATAGRAM MQMT_REQUEST MQMT_REPLY MQMT_REPORT <i>See note at the beginning of this table.</i>
IMFQ_XQH_MD_EXPIRY	Contains a decimal between 1 and 999999999 or MQEI_UNLIMITED.
IMFQ_XQH_MD_FEEDBACK	Contains one of the following codes for report messages: MQFB_NONE MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT In addition, this field can contain any reason code from the following sources: IMS-bridge feedback codes CICS-bridge feedback codes MQSeries reason codes <i>See note at the beginning of this table.</i>
IMFQ_XQH_MD_ENCODING	Contains MQENC_NATIVE or a decimal up to 999999999.
IMFQ_XQH_MD_CODEDCHARSETID	Contains MQCCSI_Q_MGR MQCCSI_EMBEDDED Or a decimal up to 999999999. <i>See IBM MQSeries Application Programming Reference Guide for details on how the CodedCharSetId field is used as input.</i>

Table C-6 Transmission Queue Header Variables – Rules and Rule-Invoked EXECs (Part 3 of 5)

Variable Name	Notes
IMFQ_XQH_MD_FORMAT	Contains one of the following formats: <ul style="list-style-type: none"> MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2 See note at the beginning of this table.
IMFQ_XQH_MD_PRIORITY	Contains a decimal between 0 and 999999999.
IMFQ_XQH_MD_PERSISTENCE	Contains one of the following values: <ul style="list-style-type: none"> MQPER_PERSISTENT MQPER_NOT_PERSISTENT
IMFQ_XQH_MD_MSGID	Contains a 32-byte MsgId. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_CORRELID	Contains a 32-byte CorrelId. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_BACKOUTCOUNT	Contains a decimal between 0 and 255.
IMFQ_XQH_MD_REPLYTOQ	Contains up to a 48-character name of the ReplyToQ.
IMFQ_XQH_MD_REPLYTOQMGR	Contains up to a 48-character name of the ReplyToQMGR.
IMFQ_XQH_MD_USERIDENTIFIER	Contains up to a 12-character UserIdentifier.
IMFQ_XQH_MD_ACCOUNTINGTOKEN	Contains MQACT_NONE or up to a 32-byte AccountingToken. It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_APPLIDENTITYDATA	Contains up to a 32-character value for ApplIdentityData.

Table C-6 Transmission Queue Header Variables – Rules and Rule-Invoked EXECs (Part 4 of 5)

Variable Name	Notes
IMFQ_XQH_MD_PUTAPPLTYPE	<p>Contains a user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> MQAT_AIX MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DOS MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <p><i>See note at the beginning of this table.</i></p>
IMFQ_XQH_MD_PUTAPPLNAME	Contains up to a 28-character value for PutApplName.
IMFQ_XQH_MD_PUTDATE	Contains an 8-character date stamp.
IMFQ_XQH_MD_PUTTIME	Contains an 8-character time stamp.
IMFQ_XQH_MD_APPLORIGINDATA	Contains a 4-character value for ApplOriginData.
IMFQ_XQH_MD_GROUPID	If a MQMD_VERSION_2 MD is present, this variable may contain a 24-byte GroupId. It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFQ_XQH_MD_MSGSEQNUMBER	If a MQMD_VERSION_2 MD is present, this variable may contain a decimal between 1 and 999999999.
IMFQ_XQH_MD_OFFSET	If a MQMD_VERSION_2 MD is present, this variable may contain a decimal between 0 and 999999999.

Table C-6 Transmission Queue Header Variables – Rules and Rule-Invoked EXECs (Part 5 of 5)

Variable Name	Notes
IMFQ_XQH_MD_MSGFLAGS	If a MQMD_VERSION_2 MD is present, this variable may contain one or more of the following character strings delimited by blanks: MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE <i>See note at the beginning of this table.</i>
IMFQ_XQH_MD_ORIGINALLENGTH	Contains a decimal between 1 and 999999999 or MQOL_UNDEFINED. <i>See note at the beginning of this table.</i>

Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs

This table lists the Message Descriptor Extension variables – Rules and Rule-Invoked EXECs:

Table C-7 Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs (Part 1 of 2)

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_MDE_STRUCID	Contains the value MQMDE_STRUC_ID.
IMFQ_MDE_VERSION	Contains the value MQMDE_VERSION_2.
IMFQ_MDE_STRUCLENGTH	Contains the value MQMDE_LENGTH_2.
IMFQ_MDE_ENCODING	Contains decimal data.
IMFQ_MDE_CODEDCHARSETID	Contains decimal data.
IMFQ_MDE_FORMAT	<p>Contains one of the following formats:</p> <ul style="list-style-type: none"> MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2 <p><i>See note at the beginning of this table.</i></p>
IMFQ_MDE_FLAGS	Contains MQMDEF_NONE.
IMFQ_MDE_GROUPID	Contains a 24-byte GroupId. No conversion of hexadecimal data is done.
IMFQ_MDE_MSGSEQNUMBER	Contains a decimal between 1 and 999999999.
IMFQ_MDE_OFFSET	Contains a decimal between 0 and 999999999.

Table C-7 Message Descriptor Extension Variables – Rules and Rule-Invoked EXECs (Part 2 of 2)

Variable Name	Notes
IMFQ_MDE_MSGFLAGS	<p>Contains one or more of the following character strings delimited by blanks:</p> <p style="padding-left: 40px;">MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFQ_MDE_ORIGINALLENGTH	<p>Contains a decimal between 1 and 999999999 or</p> <p style="padding-left: 40px;">MQOL_UNDEFINED.</p>

Work Information Header Variables – Rules and Rule-Invoked EXECs

This table lists the Work Information Header variables – Rules and Rule-Invoked EXECs:

Table C-8 Work Information Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
IMFQ_WIH_STRUCID	Contains MQWIH_STRUC_ID.
IMFQ_WIH_VERSION	Contains MQWIH_VERSION_1.
IMFQ_WIH_STRUCLENGTH	Contains MQWIH_LENGTH_1.
IMFQ_WIH_ENCODING	Contains decimal data.
IMFQ_WIH_CODEDCHARSETID	Contains decimal data.
IMFQ_WIH_FORMAT	Contains one of the following formats: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_WIH_FLAGS	Contains the value MQWIH_NONE.
IMFQ_WIH_SERVICENAME	Contains a 32-character value.
IMFQ_WIH_SERVICESTEP	Contains an 8-character value.
IMFQ_WIH_MSGTOKEN	Contains a 16-byte hexadecimal. No conversion of hexadecimal data is done.
IMFQ_WIH_RESERVED	Contains blanks.

Reference Message Header Variables – Rules and Rule-Invoked EXECs

This table lists the Reference Message Header variables – Rules and Rule-Invoked EXECs:

Table C-9 Reference Message Header Variables – Rules and Rule-Invoked EXECs (Part 1 of 2)

Variable Name	Notes
IMFQ_RMH_STRUCID	Contains the value MQRMH_STRUC_ID.
IMFQ_RMH_VERSION	Contains the value MQRMH_VERSION_1.
IMFQ_RMH_STRUCLength	Contains decimal data.
IMFQ_RMH_ENCODING	Contains decimal data.
IMFQ_RMH_CODEDCHARSETID	Contains decimal data.
IMFQ_RMH_FORMAT	Contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_RMH_FLAGS	Contains one of the following values: MQRMHF_LAST MQRMHF_NOT_LAST
IMFQ_RMH_OBJECTTYPE	Contains an 8-character value.
IMFQ_RMH_OBJECTINSTANCEID	Contains a 24-byte hexadecimal. No conversion of hexadecimal data is done.
IMFQ_RMH_SRCENVLENGTH	Contains decimal data.
IMFQ_RMH_SRCENVOFFSET	Contains decimal data.
IMFQ_RMH_SRCNAMELENGTH	Contains decimal data.
IMFQ_RMH_SRCNAMEOFFSET	Contains decimal data.

Table C-9 Reference Message Header Variables – Rules and Rule-Invoked EXECs (Part 2 of 2)

Variable Name	Notes
IMFQ_RMH_DESTENVLENGTH	Contains decimal data.
IMFQ_RMH_DESTENVOFFSET	Contains decimal data.
IMFQ_RMH_DESTNAMELENGTH	Contains decimal data.
IMFQ_RMH_DESTNAMEOFFSET	Contains decimal data.
IMFQ_RMH_DATALOGICALENGTH	Contains decimal data.
IMFQ_RMH_DATALOGICALOFFSET	Contains decimal data.
IMFQ_RMH_DATALOGICALOFFSET2	Contains decimal data.

Trigger Message 2 (Character Format) Variables – Rules and Rule-Invoked EXECs

This table lists the trigger message 2 (character format) variables – Rules and Rule-Invoked EXECs:

Table C-10 **Trigger Message 2 (Character format) Variables – Rules and Rule-Invoked EXECs**

Variable Name	Notes
IMFQ_TMC_STRUCID	Contains MQTMC_STRUC_ID.
IMFQ_TMC_VERSION	Contains MQTMC_VERSION_1.
IMFQ_TMC_QNAME	Contains 48-character name of triggered queue.
IMFQ_TMC_PROCESSNAME	Contains 48-character name of process object.
IMFQ_TMC_TRIGGERDATA	Contains 64-character free-format trigger data.
IMFQ_TMC_APPLTYPE	Contains 4 blanks.
IMFQ_TMC_APPLID	Contains a 256-character string.
IMFQ_TMC_ENVDATA	Contains a 128-character string.
IMFQ_TMC_USERDATA	Contains a 128-character string.
IMFQ_TMC_QMGRNAME	Contains a 48-character value.

Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs

This table lists the Rules and formatting header variables – Rules and Rule-Invoked EXECs:

Table C-11 Rules and Formatting Header Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_RFH_STRUCID	Contains MQRFH_STRUC_ID.
IMFQ_RFH_VERSION	Contains MQRFH_VERSION_1
IMFQ_RFH_STRUCLENGTH	Contains decimal data.
IMFQ_RFH_ENCODING	Contains decimal data or MQENC_NATIVE.
IMFQ_RFH_CODEDCHARSETID	Contains decimal data.
IMFQ_RFH_FORMAT	Contains one of the following formats: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_RFH_FLAGS	Contains following character string delimited by blanks: MQRFH_NONE. <i>See note at the beginning of this table.</i>
IMFQ_RFH_NAMEVALUESTRING	Contains a character string that is the length of IMFQ_RFH_STRUCLENGTH minus MQRMH_STRUC_LENGTH_FIXED. <i>See the IBM MQSeries Application Programming Reference Guide for information about these constants.</i>

Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs

This table lists the Rules and formatting header variables version 2 – Rules and Rule-Invoked EXEC:

Table C-12 Rules and Formatting Header Variables Version 2 – Rules and Rule-Invoked EXECs

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_RFH2_STRUCID	Contains the constant MQRFH_STRUC_ID.
IMFQ_RFH2_VERSION_2	Contains the constant MQRFH_VERSION_2.
IMFQ_RFH2_STRUCLength	Contains decimal data.
IMFQ_RFH2_ENCODING	Contains decimal data or the constant MQENC_NATIVE.
IMFQ_RFH2_CODEDCHARSETID	Contains decimal data.
IMFQ_RFH2_FORMAT	Contains one of the following formats: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFQ_RFH2_FLAGS	Contains the following character string delimited by blanks: MQRFH_NONE. <i>See note at the beginning of this table.</i>
IMFQ_RFH2_NAMEVALUECCSID	Contains decimal data.
IMFQ_RFH2_NAMEVALUEDATA	Contains the exact string of data contained in the message buffer in the position following the NameValueCCSID field.

PCF Variables – Rules and Rule-Invoked EXECs

This table lists the PCF variables – Rules and Rule-Invoked EXECs:

Table C-13 PCF Variables – Rules and Rule-Invoked EXECs

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFQ_CFH_TYPE	Contains one or more of the following character strings delimited by blanks: MQCFT_EVENT MQCFT_COMMAND <i>See note at the beginning of this table.</i>
IMFQ_CFH_STRUCLength	Contains decimal data.
IMFQ_CFH_VERSION	Contains the constant MQCFH_VERSION_1.
IMFQ_CFH_COMMAND	Contains one of the following constants: MQCMD_Q_MGR_EVENT MQCMD_PERFM_EVENT MQCMD_CHANNEL_EVENT <i>See note at the beginning of this table.</i>
IMFQ_CFH_MSGSEQNUMBER	Contains a decimal.
IMFQ_CFH_CONTROL	Contains the constant MQCFC_LAST.
IMFQ_CFH_COMPCODE	Contains one of the following constants: MQCC_OK MQCC_WARNING <i>See note at the beginning of this table.</i>
IMFQ_CFH_REASON	Contains a numeric reason code between 0 and 999999999.
IMFQ_CFH_PARAMETERCOUNT	Contains decimal data.

Appendix D EXECs Variables

General Purpose Variables – EXECs

This table lists the general purpose variable:

Variable Name	Data Type	Notes
IMFMQI_STRUCTURES	Character	Contains names of all structures in a message, for example: MD XQH.
IMFMQI_OFFSETS	Character	Contains a blank-delimited list of offsets to structures and application data from the beginning of the message buffer. For example: if the message buffer contains a Dead Letter Header (DLH) and application data, the variable value may look like this: 'NONE 0 172', indicating NONE for MD (since it is not in the message buffer), 0 for the DLH (first structure in the buffer) and 172 for application data (follows the DLH).
IMFMQI_REASON	Character	Contains the constant reason code from the IMFEXEC MQI statement and the IMFEXEC CMD ... TYPE(MQS) command. For example, if IMFMQRC = 2033, IMFMQI_REASON = 'MQRC_NO_MSG_AVAILABLE'

Options and Miscellaneous Variables

This table lists the options and miscellaneous variables:

Variable	Description	Input/ Output	Call Type(s)
IMFMQI_CO_OPTIONS	CLOSE options	Input	CLOSE
IMFMQI_GMO_OPTIONS	GET options	Input	GET
IMFMQI_GMO_STRUCID	optional GET options The EXEC does not need to set this variable. If the EXEC does not set the variable, the MQSeries default, 'MQGMO_STRUC_ID', is used unmodified by the EXEC Manager.	Input	GET
IMFMQI_GMO_VERSION	optional GET options The valid value for this variable is 'MQGMO_VERSION_1', which is also the default.	Input	GET
IMFMQI_GMO_OPTIONS	optional GET options Valid values are MQGMO_CONVERT MQGMO_FAIL_IF QUIESCING MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_MSG_UNDER_CURSOR MQGMO_MARK_SKIP_BACKOUT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_BROWSE_NEXT MQGMO_BROWSE_FIRST MQGMO_SET_SIGNAL MQGMO_NO_SYNCPOINT MQGMO_SYNCPOINT MQGMO_WAIT MQGMO_NO_WAIT MQGMO_NONE Note that this list might not be complete, as IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> . Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, IMFMQI_GMO_OPTIONS = 'MQGMO_SYNCPOINT MQGMO_CONVERT'	Input	GET
IMFMQI_GMO_WAITINTERVAL	optional GET options Specify a decimal number in the range 0 to 999999999.	Input	GET

Variable	Description	Input/ Output	Call Type(s)
IMFMQI_GMO_RESOLVED_QMGRNAME	GET options Is the resolved name of the destination queue manager.	Output	GET
IMFMQI_GMO_RESOLVED_QNAME	GET options Is the resolved name of the destination queue.	Output	GET
IMFMQI_PMO_STRUCID	optional PUT options The EXEC does not need to set this variable. If the EXEC does not set the variable, the MQSeries default ('MQPMO_STRUC_ID'), which is unmodified by the EXEC manager is used.	Input	PUT, PUT1
IMFMQI_PMO_VERSION	optional PUT options The valid value for this variable is 'MQPMO_VERSION_1', which is also the default.	Input	PUT, PUT1
IMFMQI_PMO_OPTIONS	optional PUT options Valid values are MQPMO_SYNCPOINT MQPMO_NO_SYNCPOINT MQPMO_NEW_MSG_ID MQPMO_NEW_CORREL_ID MQPMO_LOGICAL_ORDER MQPMO_NO_CONTEXT MQPMO_DEFAULT_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT MQPMO_PASS_ALL_CONTEXT MQPMO_SET_IDENTITY_CONTEXT MQPMO_SET_ALL_CONTEXT MQPMO_ALTERNATE_USER_AUTHORITY MQPMO_FAIL_IF QUIESCING MQPMO_NONE Note that this list might not be complete, as IBM might have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> . Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, IMFMQI_PMO_OPTIONS = 'MQPMO_SYNCPOINT MQPMO_PASS_ALL_CONTEXT'	Input	PUT, PUT1
IMFMQI_PMO_TIMEOUT	optional PUT options Specify a decimal number in the range 0 to 999999999.	Input	PUT, PUT1

Variable	Description	Input/ Output	Call Type(s)
IMFMQI_PMO_CONTEXT	optional PUT options If 'MQPMO_PASS_IDENTITY_CONTEXT" or 'MQPMO_PASS_ALL_CONTEXT" is specified for IMFMQI_PMO_OPTIONS, the value must be the object handle of an input queue from which context information is taken.	Input	PUT, PUT1
IMFMQI_PMO_RESOLVED QNAME	PUT options Is the resolved name of the destination queue.	Output	PUT, PUT1
IMFMQI_PMO_RESOLVED QMGRNAME	PUT options Is the resolved name of the destination queue manager.	Output	PUT, PUT1
IMFMQI_OO_OPTIONS	optional OPEN options Valid values are MQOO_BIND_NOT_FIXED MQOO_BIND_ON_OPEN MQOO_FAIL_IF QUIESCING MQOO_ALTERNATE_USER_AUTHORITY MQOO_SET_ALL_CONTEXT MQOO_SET_IDENTITY_CONTEXT MQOO_PASS_ALL_CONTEXT MQOO_PASS_IDENTITY_CONTEXT MQOO_SAVE_ALL_CONTEXT MQOO_SET MQOO_INQUIRE MQOO_OUTPUT MQOO_BROWSE MQOO_INPUT_EXCLUSIVE MQOO_INPUT_SHARED MQOO_INPUT_AS_Q_DEF MQOO_BIND_AS_Q_DEF Note that this list might not be complete, as IBM might have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> . Also note that when specifying multiple options, they must be enclosed within single quotation marks. For example, IMFMQI_OO_OPTIONS = 'MQOO_FAIL_IF QUIESCING MQOO_INPUT_SHARED'	Input	OPEN
IMFMQI_BUFFER	message buffer	Input	GET, PUT, PUT1

Variable	Description	Input/ Output	Call Type(s)
IMFMQI_BUFFLEN	message buffer length	Input	GET, PUT, PUT1
IMFMQI_DATALEN	length of returned message	Output	GET

EVENT Variables – EXECs

This table lists the event variables - EXECs:

Table D-1 Event Variables - EXECs (Part 1 of 3)

Variable Name	Notes
IMFMQI_EVENT_TYPE	event type-not a MQSeries field
IMFMQI_EVENT_QMGRNAME	name of queue manager that generated event, whether local or remote
IMFMQI_EVENT_QNAME	name of queue from Object Descriptor
IMFMQI_EVENT_BASEQNAME	name of queue manager (to which the alias resolves)
IMFMQI_EVENT_APPLNAME	name of application
IMFMQI_EVENT_OBJECTQMGRNAME	name of the object queue manager
IMFMQI_EVENT_CHANNEL_NAME	name of channel
IMFMQI_EVENT_CONNECTIONNAME	name of connection. For TCP/IP, this is the internet address when the channel has established a connection successfully Otherwise, it is the contents of the ConnectionName field in the channel definition.
IMFMQI_EVENT_XMITQNAME	transmission queue name

Table D-1 Event Variables - EXECs (Part 2 of 3)

Variable Name	Notes
IMFMQI_EVENT_FORMAT	one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_EVENT_QTYPE	type of queue
IMFMQI_EVENT_APPLTYPE	type of queue to which the alias resolves (A for alias, M for model)
IMFMQI_EVENT_REASONQUALIFIER	reason qualifier: MQRQ_CHANNEL_STOPPED_OK MQRQ_CHANNEL_STOPPED_ERROR MQRQ_CHANNEL_STOPPED_RETRY MQRQ_CHANNEL_STOPPED_DISABLED
IMFMQI_EVENT_ERRORIDENTIFIER	See the section regarding the CHANNEL_STOPPED event, field Error identifier, in the IBM publication <i>MQSeries Programmable System Management</i> for a more detail description.
IMFMQI_EVENT_ERRORMSGID	last three characters of IMFMQI_EVENT_ERRORIDENTIFIER (ErrorIdentifier) Use this number to look up the message ID in the "Distributed Queuing Message Codes" section of the <i>IBM MQSeries for MVS/ESA Messages and Codes</i> manual.
IMFMQI_EVENT_PROCESSNAME	name of process
IMFMQI_EVENT_USERIDENTIFIER	name of user identifier
IMFMQI_EVENT_AUXERRORDATAINT1	first integer of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATAINT2	second integer of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATASTR1	first string of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATASTR2	second string of auxiliary error data for channel errors
IMFMQI_EVENT_AUXERRORDATASTR3	third string of auxiliary error data for channel errors

Table D-1 Event Variables - EXECs (Part 3 of 3)

Variable Name	Notes
IMFMQI_EVENT_TIMESINCERESET	time (in seconds) since the statistics were last reset; for this event, this is greater than the service interval
IMFMQI_EVENT_HIGHQDEPTH	max number of message on the queue since the statistics were last reset
IMFMQI_EVENT_MSGENQCOUNT	number of messages enqueued since statistics were last reset
IMFMQI_EVENT_MSGDEQCOUNT	number of messages dequeued since statistics were last reset
IMFMQI_EVENT_BRIDGENAME	name of bridge
IMFMQI_EVENT_BRIDGETYPE	only current value is 1 (OTMA)
IMFMQI_EVENT_OPTIONS	open options
IMFMQI_EVENT_COMMAND	name of command event
IMFMQI_EVENT_CURDEPTH	current depth of the event queue
IMFMQI_EVENT_MAXDEPTH	maximum depth of the event queue
IMFMQI_EVENT_PERCENTFULL	percent full of the event queue

Dead Letter Header Variables – EXECs

This table lists the Dead Letter Header variables - EXECs:

Table D-2 Dead Letter Header Variables - EXECs (Part 1 of 2)

Variable Name	Notes
Note: This list may not be complete because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFMQI_DLH_STRUCID	contains the value MQDLH_STRUC_ID
IMFMQI_DLH_VERSION	contains the value MQDLH_VERSION_1
IMFMQI_DLH_REASON	contains a numeric reason code between 0 and 999999999
IMFMQI_DLH_DESTQNAME	contains the 48-character destination queue name
IMFMQI_DLH_DESTQMGRNAME	contains the 48-character destination queue manager name
IMFMQI_DLH_ENCODING	contains decimal data
IMFMQI_DLH_CODEDCHARSETID	contains decimal data
IMFMQI_DLH_FORMAT	contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2

Table D-2 Dead Letter Header Variables - EXECs (Part 2 of 2)

Variable Name	Notes
IMFMQI_DLH_PUTAPPLTYPE	<p>contains a numeric user-defined type or one of the following standard types:</p> <ul style="list-style-type: none"> MQAT_AIX MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DOS MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <p><i>See note at the beginning of this table.</i></p>
IMFMQI_DLH_PUTAPPLNAME	contains up to a 28-character name for PutApplName
IMFMQI_DLH_PUTDATE	contains an eight-character date stamp
IMFMQI_DLH_PUTTIME	contains an eight-character time stamp

Trigger Message Variables – EXECs

This table lists the trigger message variables - EXECs:

Table D-3 **Trigger Message Variables - EXECs**

Variable Name	Data Type	Notes
Note: This this list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .		
IMFMQI_TM_STRUCID	Character	contains the value MQTM_STRUC_ID
IMFMQI_TM_VERSION	Decimal	contains the value MQTM_VERSION_1
IMFMQI_TM_QNAME	Character	contains 48-character name of the triggered queue
IMFMQI_TM_PROCESSNAME	Character	contains 48-character name of the process object
IMFMQI_TM_TRIGGERDATA	Character	contains 64-character free-format trigger data
IMFMQI_TM_APPLTYPE	Decimal	contains a numeric user-defined type or one of the following standard types: MQAT_AIX MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DOS MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <i>See note at the beginning of this table.</i>
IMFMQI_TM_APPLID	Character	contains a 256-character string
IMFMQI_TM_ENVDATA	Character	contains a 128-character string
IMFMQI_TM_USERDATA	Character	contains a 128-character string

CICS Information Header Variables – EXECs

This table lists the CICS Information Header variables - EXECs:

Table D-4 CICS Information Header Variables - EXECs (Part 1 of 4)

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFMQI_CIH_STRUCID	contains MQCIH_STRUC_ID
IMFMQI_CIH_VERSION	contains MQCIH_VERSION_2
IMFMQI_CIH_STRUCLENGTH	contains MQCIH_LENGTH_2
IMFMQI_CIH_ENCODING	contains the decimal 0
IMFMQI_CIH_CODEDCHARSETID	contains the decimal 0
IMFMQI_CIH_FORMAT	contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_CIH_FLAGS	contains MQCIH_NONE
IMFMQI_CIH_RETURNCODE	contains one of the following values: MQCRC_OK MQCRC_CICS_EXEC_ERROR MQCRC_MQ_API_ERROR MQCRC_BRIDGE_ERROR MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_BRIDGE_TIMEOUT MQCRC_TRANSID_NOT_AVAILABLE

Table D-4 CICS Information Header Variables - EXECs (Part 2 of 4)

Variable Name	Notes
IMFMQI_CIH_COMPCODE	contains decimal data
IMFMQI_CIH_REASON	contains decimal data
IMFMQI_CIH_UOWCONTROL	contains one of the following values: MQCUOWC_ONLY MQCUOWC_CONTINUE MQCUOWC_FIRST MQCUOWC_MIDDLE MQCUOWC_LAST MQCUOWC_COMMIT MQCUOWC_BACKOUT
IMFMQI_CIH_GETWAITINTERVAL	contains decimal data or one of the following constants: MQCGWI_DEFAULT MQWI_UNLIMITED
IMFMQI_CIH_LINKTYPE	contains one of the following values: MQCLT_PROGRAM MQCLT_TRANSACTION
IMFMQI_CIH_OUTPUTDATALENGTH	contains decimal data or the constant MQCLT_TRANSACTION
IMFMQI_CIH_FACILITYKEEPTIME	contains decimal data
IMFMQI_CIH_ADSDDESCRIPTOR	contains one of the following values: MQCADSD_NONE MQCADSD_SEND MQCADSD_RECV MQCADSD_MSGFORMAT <i>See note at the beginning of this table.</i>
IMFMQI_CIH_CONVERSATIONALTASK	contains one of the following values: MQCCT_YES MQCCT_NO
IMFMQI_CIH_TASKENDSTATUS	contains one of the following values: MQCTES_NOSYNC MQCTES_COMMIT MQCTES_BACKOUT MQCTES_ENDTASK
IMFMQI_CIH_FACILITY	contains a eight-byte hexadecimal value or the constant MQCFAC_NONE No conversion of hexadecimal data is done.

Table D-4 CICS Information Header Variables - EXECs (Part 3 of 4)

Variable Name	Notes
IMFMQI_CIH_FUNCTION	contains a four-character value of one of the following constants: MQCFUNC_MQCONN MQCFUNC_MQGET MQCFUNC_MQINQ MQCFUNC_MQOPEN MQCFUNC_MQPUT MQCFUNC_MQPUT1 MQCFUNC_NONE <i>See note at the beginning of this table.</i>
IMFMQI_CIH_ABENDCODE	contains a four-character value
IMFMQI_CIH_AUTHENTICATOR	contains an eight-character value
IMFMQI_CIH_RESERVED1	contains eight blanks
IMFMQI_CIH_REPLYTOFORMAT	contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_CIH_REMOTESYSID	contains four blanks
IMFMQI_CIH_REMOTETRANSID	contains four blanks
IMFMQI_CIH_TRANSACTIONID	contains a four-character value
IMFMQI_CIH_FACILITYLIKE	contains a four-character value
IMFMQI_CIH_ATTENTIONID	contains a four-character value

Table D-4 CICS Information Header Variables - EXECs (Part 4 of 4)

Variable Name	Notes
IMFMQI_CIH_STARTCODE	contains one of the following constants: MQCSC_START MQCSC_STARTDATA MQCSC_TERMINPUT MQCSC_NONE <i>See note at the beginning of this table.</i>
IMFMQI_CIH_CANCELCODE	contains a four-character value
IMFMQI_CIH_NEXTTRANSACTIONID	contains a four-character value
IMFMQI_CIH_RESERVED2	contains eight blanks
IMFMQI_CIH_RESERVED3	contains eight blanks
IMFMQI_CIH_CURSORPOSITION	contains decimal data
IMFMQI_CIH_ERROROFFSET	contains decimal data
IMFMQI_CIH_INPUTITEM	contains the decimal 0
IMFMQI_CIH_RESERVED4	contains the decimal 0

IMS Information Header Variables – EXECs

This table lists the IMS Information Header variables - EXECs:

Table D-5 IMS Information Header Variables - EXECs (Part 1 of 2)

Variable Name	Notes
IMFMQI_IIH_STRUCID	contains MQIIH_STRUC_ID
IMFMQI_IIH_VERSION	contains MQIIH_VERSION_1
IMFMQI_IIH_STRUCLength	contains MQIIH_LENGTH_1
IMFMQI_IIH_ENCODING	contains the decimal 0
IMFMQI_IIH_CODEDCHARSETID	contains the decimal 0
IMFMQI_IIH_FORMAT	contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_IIH_FLAGS	contains MQIIH_NONE
IMFMQI_IIH_LTERM_OVERRIDE	contains an eight-character value
IMFMQI_IIH_MFSMAPNAME	contains an eight-character value

Table D-5 IMS Information Header Variables - EXECs (Part 2 of 2)

Variable Name	Notes
IMFMQI_IIH_REPLYTOFORMAT	contains one of the following format constants: MQFMT_NONE (IBM default) MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PC MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_IIH_AUTHENTICATOR	contains an 8-character value or the value MQIAUT_NONE
IMFMQI_IIH_TRANINSTANCEID	contains a 16-byte hexadecimal value or the constant MQITII_NONE No conversion of hexadecimal data is done.
IMFMQI_IIH_TRANSTATE	contains one of the following constants: MQITS_IN_CONVERSATION MQITS_NOT_IN_CONVERSATION MQITS_ARCHITECTED
IMFMQI_IIH_COMMITMODE	contains one of the following constants: MQICM_COMMIT_THEN_SEND MQICM_SEND_THEN_COMMIT
IMFMQI_IIH_SECURITYSCOPE	contains one of the following constants: MQISS_CHECK MQISS_FULL
IMFMQI_IIH_RESERVED	contains 1 blank

Transmission Queue Header Variables – EXECs

This table lists the Transmission Queue Header variables - EXECs:

Table D-6 **Transmission Queue Header Variables - EXECs (Part 1 of 5)**

Variable Name	Notes
<p>Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>	
IMFMQI_XQH_STRUCID	contains MQXQH_STRUC_ID
IMFMQI_XQH_VERSION	contains MQXQH_VERSION_1
IMFMQI_XQH_REMOTEQNAME	contains a 48-character value
IMFMQI_XQH_REMOTEQMGRNAME	contains a 48-character value
IMFMQI_XQH_MD_STRUCID	contains MQMD_STRUC_ID
IMFMQI_XQH_MD_VERSION	contains MQMD_VERSION_1
IMFMQI_XQH_MD_REPORT	<p>contains one or more of the following delimited by blanks:</p> <p>MQRO_EXCEPTION MQRO_EXCEPTION_WITH_DATA MQRO_EXCEPTION_WITH_FULL_DATA MQRO_EXPIRATION MQRO_EXPIRATION_WITH_DATA MQRO_EXPIRATION_WITH_FULL_DATA MQRO_COA MQRO_COA_WITH_DATA MQRO_COA_WITH_FULL_DATA MQRO_COD MQRO_COD_WITH_DATA MQRO_COD_WITH_FULL_DATA MQRO_PAN MQRO_NAN MQRO_NEW_MSG_ID MQRO_PASS_MSG_ID MQRO_COPY_MSG_ID_TO_CORREL_ID MQRO_PASS_CORREL_ID MQRO_DEAD_LETTER_Q MQRO_DISCARD_MSG MQRO_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_MSGTYPE	<p>contains one of the following values:</p> <p>MQMT_DATAGRAM MQMT_REQUEST MQMT_REPLY MQMT_REPORT</p> <p><i>See note at the beginning of this table.</i></p>

Table D-6 Transmission Queue Header Variables - EXECs (Part 2 of 5)

Variable Name	Notes
IMFMQI_XQH_MD_EXPIRY	contains a decimal between 1 and 999999999, or MQEI_UNLIMITED
IMFMQI_XQH_MD_FEEDBACK	<p>contains one of the following values for report messages:</p> <p style="padding-left: 40px;">MQFB_NONE MQFB_COA MQFB_COD MQFB_EXPIRATION MQFB_PAN MQFB_NAN MQFB_QUIT</p> <p>In addition, this field can contain a reason code from the following sources:</p> <ul style="list-style-type: none"> • IMS-bridge feedback codes • CICS-bridge feedback codes • MQSeries reason codes <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_ENCODING	contains MQENC_NATIVE or any decimal up to 999999999.
IMFMQI_XQH_MD_CODEDCHARSETID	<p>contains</p> <p style="padding-left: 40px;">MQCCSI_Q_MGR MQCCSI_EMBEDDED or a decimal up to 999999999</p> <p><i>See the IBM MQSeries Application Programming Reference Guide for details about how the CodedCharSetId field is used as input.</i></p>

Table D-6 Transmission Queue Header Variables - EXECs (Part 3 of 5)

Variable Name	Notes
IMFMQI_XQH_MD_FORMAT	contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2 <i>See note at the beginning of this table.</i>
IMFMQI_XQH_MD_PRIORITY	contains a decimal between 0 and 999999999
IMFMQI_XQH_MD_PERSISTENCE	contains one of the following values: MQPER_PERSISTENT MQPER_NOT_PERSISTENT
IMFMQI_XQH_MD_MSGID	contains a 32-byte Msgld. No conversion of hexadecimal data is done
IMFMQI_XQH_MD_CORRELID	contains a 32-byte Correlld. No conversion of hexadecimal data is done
IMFMQI_XQH_MD_BACKOUTCOUNT	contains a number between 0 and 255
IMFMQI_XQH_MD_REPLYTOQ	contains up to a 48-character name of the ReplyToQ
IMFMQI_XQH_MD_REPLYTOQMGR	contains up to a 48-character name of the ReplyToQMgr
IMFMQI_XQH_MD_USERIDENTIFIER	contains up to a 12-character UserIdentifier
IMFMQI_XQH_MD_ACCOUNTINGTOKEN	contains MQACT_NONE or up to a 32-byte AccountingToken It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_XQH_MD_APPLIDENTITYDATA	contains up to a 32-character value for ApplIdentityData

Table D-6 Transmission Queue Header Variables - EXECs (Part 4 of 5)

Variable Name	Notes
IMFMQI_XQH_MD_PUTAPPLTYPE	contains a numeric user-defined type or one of the following standard types: MQAT_AIX MQAT_CICS MQAT_CICS_BRIDGE MQAT_CICS_VSE MQAT_DOS MQAT_GUARDIAN MQAT_IMS MQAT_IMS_BRIDGE MQAT_MVS MQAT_NOTES_AGENT MQAT_NSK MQAT_OS2 MQAT_OS390 MQAT_OS400 MQAT_QMGR MQAT_UNIX MQAT_VMS MQAT_VOS MQAT_WINDOWS MQAT_WINDOWS_NT MQAT_XCF MQAT_UNKNOWN <i>See note at the beginning of this table.</i>
IMFMQI_XQH_MD_PUTAPPLNAME	contains up to a 28-character value for PutApplName
IMFMQI_XQH_MD_PUTDATE	contains an eight-character date stamp
IMFMQI_XQH_MD_PUTTIME	contains an eight-character time stamp
IMFMQI_XQH_MD_APPLORIGINDATA	contains a four-character value for ApplOriginData
IMFMQI_XQH_MD_GROUPID	if a MQMD_VERSION_2 MD is present, might contain a 24-byte GroupId It is processed exactly as received from the GET. No conversion of hexadecimal data is done.
IMFMQI_XQH_MD_MSGSEQNUMBER	if a MQMD_VERSION_2 MD is present, might contain a decimal between 1 and 999999999
IMFMQI_XQH_MD_OFFSET	if a MQMD_VERSION_2 MD is present, might contain a decimal between 0 and 999999999

Table D-6 **Transmission Queue Header Variables - EXECs (Part 5 of 5)**

Variable Name	Notes
IMFMQI_XQH_MD_MSGFLAGS	<p>if a MQMD_VERSION_2 MD is present, might contain one or more of the following character strings delimited by blanks:</p> <p style="padding-left: 40px;">MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_XQH_MD_ORIGINALLENGTH	<p>contains a decimal between 1 and 999999999 or MQOL_UNDEFINED</p> <p><i>See note at the beginning of this table.</i></p>

Message Descriptor Extension Variables – EXECs

This table lists the Message Descriptor Extension variables - EXECs:

Table D-7 Message Descriptor Extension Variables - EXECs (Part 1 of 2)

Variable Name	Notes
<p>Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i>.</p>	
IMFMQI_MDE_STRUCID	contains MQMDE_STRUC_ID
IMFMQI_MDE_VERSION	contains MQMDE_VERSION_2
IMFMQI_MDE_STRUCLength	contains MQMDE_LENGTH_2
IMFMQI_MDE_ENCODING	contains decimal data
IMFMQI_MDE_CODEDCHARSETID	contains decimal data
IMFMQI_MDE_FORMAT	<p>contains one of the following format constants:</p> <ul style="list-style-type: none"> MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_MDE_FLAGS	contains MQMDEF_NONE
IMFMQI_MDE_GROUPID	contains a 24-byte GroupId. No conversion of hexadecimal data is done
IMFMQI_MDE_MSGSEQNUMBER	contains a decimal between 1 and 999999999
IMFMQI_MDE_OFFSET	contains a decimal between 0 and 999999999

Table D-7 Message Descriptor Extension Variables - EXECs (Part 2 of 2)

Variable Name	Notes
IMFMQI_MDE_MSGFLAGS	<p>contains one or more of the following character strings delimited by blanks:</p> <p style="padding-left: 40px;">MQMF_SEGMENTATION_INHIBITED MQMF_SEGMENTATION_ALLOWED MQMF_MSG_IN_GROUP MQMF_LAST_MSG_IN_GROUP MQMF_SEGMENT MQMF_LAST_SEGMENT MQMF_NONE.</p> <p><i>See note at the beginning of this table.</i></p>
IMFMQI_MDE_ORIGINALLENGTH	<p>contains a decimal between 1 and 999999999 or MQOL_UNDEFINED</p>

Work Information Header Variables – EXECs

This table lists the Work Information Header variables - EXECs:

Table D-8 Work Information Header Variables - EXECs

Variable Name	Notes
IMFMQI_WIH_STRUCID	contains the value MQWIH_STRUC_ID
IMFMQI_WIH_VERSION	contains the value MQWIH_VERSION_1
IMFMQI_WIH_STRUCLength	contains the value MQWIH_LENGTH_1
IMFMQI_WIH_ENCODING	contains decimal data
IMFMQI_WIH_CODEDCHARSETID	contains decimal data
IMFMQI_WIH_FORMAT	contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_WIH_FLAGS	contains the value MQWIH_NONE
IMFMQI_WIH_SERVICENAME	contains a 32-character value
IMFMQI_WIH_SERVICESTEP	contains an 8-character value
IMFMQI_WIH_MSGTOKEN	contains a 16-byte hexadecimal value No conversion of hexadecimal data is done.
IMFMQI_WIH_RESERVED	contains blanks

Reference Message Header Variables – EXECs

This table lists the Reference Message Header variables - EXECs:

Table D-9 Reference Message Header Variables - EXECs (Part 1 of 2)

Variable Name	Note
IMFMQI_RMH_STRUCID	contains MQRMH_STRUC_ID
IMFMQI_RMH_VERSION	contains MQRMH_VERSION_1
IMFMQI_RMH_STRUCLENGTH	contains decimal data
IMFMQI_RMH_ENCODING	contains decimal data
IMFMQI_RMH_CODEDCHARSETID	contains decimal data
IMFMQI_RMH_FORMAT	contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_RMH_FLAGS	contains one of the following values: MQRMHF_LAST MQRMHF_NOT_LAST
IMFMQI_RMH_OBJECTTYPE	contains an 8-character value
IMFMQI_RMH_OBJECTINSTANCEID	contains a 24-byte hexadecimal value No conversion of hexadecimal data is done.
IMFMQI_RMH_SRCENVLENGTH	contains decimal data
IMFMQI_RMH_SRCENVOFFSET	contains decimal data
IMFMQI_RMH_SRCNAMELENGTH	contains decimal data
IMFMQI_RMH_SRCNAMEOFFSET	contains decimal data
IMFMQI_RMH_DESTENVLENGTH	contains decimal data

Table D-9 Reference Message Header Variables - EXECs (Part 2 of 2)

Variable Name	Note
IMFMQI_RMH_DESTENVOFFSET	contains decimal data
IMFMQI_RMH_DESTNAMELENGTH	contains decimal data
IMFMQI_RMH_DESTNAMEOFFSET	contains decimal data
IMFMQI_RMH_DATALOGICALENGTH	contains decimal data
IMFMQI_RMH_DATALOGICALOFFSET	contains decimal data
IMFMQI_RMH_DATALOGICALOFFSET2	contains decimal data

Trigger Message 2 (Character Format) Variables – EXECs

This table lists the trigger message 2 (character format) variables – EXECs:

Table D-10 **Trigger Message 2 (Character format) Variables – EXECs**

Variable Name	Notes
IMFMQI_TMC_STRUCID	contains MQTMC_STRUC_ID
IMFMQI_TMC_VERSION	contains MQTMC_VERSION_1
IMFMQI_TMC_QNAME	contains 48-character name of the triggered queue
IMFMQI_TMC_PROCESSNAME	contains 48-character name of the process object
IMFMQI_TMC_TRIGGERDATA	contains 64-character free-format trigger data
IMFMQI_TMC_APPLTYPE	contains four blanks
IMFMQI_TMC_APPLID	contains a 256-character string
IMFMQI_TMC_ENVDATA	contains a 128-character string
IMFMQI_TMC_USERDATA	contains a 128-character string
IMFMQI_TMC_QMGRNAME	contains a 48-character value

Rules and Formatting Header Variables – EXECs

This table lists the Rules and formatting header variables – EXECs:

Table D-11 Rules and Formatting Header Variables – EXECs

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFMQI_RFH_STRUCID	contains MQRFH_STRUC_ID
IMFMQI_RFH_VERSION	contains MQRFH_VERSION_1
IMFMQI_RFH_STRUCLength	contains decimal data
IMFMQI_RFH_ENCODING	contains decimal data or MQENC_NATIVE
IMFMQI_RFH_CODEDCHARSETID	contains decimal data
IMFMQI_RFH_STRUCID	contains MQRFH_STRUC_ID
IMFMQI_RFH_FORMAT	contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_RFH_FLAGS	contains the following character string delimited by blanks: MQRFH_NONE See note at the beginning of this table.
IMFMQI_RFH_NAMEVALUESTRING	contains a character string of the length of IMFQ_RFH_STRUCLength minus MQRMH_STRUC_LENGTH_FIXED See note at the beginning of this table.

Rules and Formatting Header Variables Version 2 – EXECs

This table lists the Rules and formatting header variables version 2 – EXECs:

Table D-12 Rules and Formatting Header Variables Version 2 – EXECs

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFMQI_RFH2_STRUCID	contains MQRFH_STRUC_ID
IMFMQI_RFH2_VERSION_2	contains MQRFH_VERSION_2
IMFMQI_RFH2_STRUCLength	contains decimal data
IMFMQI_RFH2_ENCODING	contains decimal data or MQENC_NATIVE
IMFMQI_RFH2_CODEDCHARSETID	contains decimal data
IMFMQI_RFH2_FORMAT	contains one of the following format constants: MQFMT_NONE MQFMT_ADMIN MQFMT_CHANNEL_COMPLETED MQFMT_CICS MQFMT_COMMAND_1 MQFMT_COMMAND_2 MQFMT_DEAD_LETTER_HEADER MQFMT_EVENT MQFMT_IMS MQFMT_IMS_VAR_STRING MQFMT_MD_EXTENSION MQFMT_PCF MQFMT_REF_MSG_HEADER MQFMT_STRING MQFMT_TRIGGER MQFMT_WORK_INFO_HEADER MQFMT_XMIT_Q_HEADER MQFMT_RF_HEADER MQFMT_RF_HEADER_2
IMFMQI_RFH2_FLAGS	contains the following character string delimited by blanks: MQRFH_NONE See note at the beginning of this table.
IMFMQI_RFH2_NAMEVALUECCSID	contains decimal data
IMFMQI_RFH2_NAMEVALUEDATA	contains the exact string of data that is contained in the message buffer in the position following the NameValueCCSID field

PCF Variables – EXECs

This table lists the PCF variables – EXECs:

Table D-13 PCF Variables – (Rules and Rule-Invoked EXECs)

Variable Name	Notes
Note: This list may not be complete, because IBM may have added new constants since the printing of this manual. For a more complete list, see the <i>IBM MQSeries Application Programming Reference Guide</i> .	
IMFMQI_CFH_TYPE	contains one or more of the following character strings delimited by blanks: MQCFT_EVENT MQCFT_COMMAND See note at the beginning of this table.
IMFMQI_CFH_STRUCLength	contains decimal data
IMFMQI_CFH_VERSION	contains MQCFH_VERSION_1
IMFMQI_CFH_COMMAND	contains one of the following constants: MQCMD_Q_MGR_EVENT MQCMD_PERFM_EVENT MQCMD_CHANNEL_EVENT See note at the beginning of this table.
IMFMQI_CFH_MSGSEQNUMBER	contains decimal data
IMFMQI_CFH_CONTROL	contains the constant MQCFC_LAST
IMFMQI_CFH_COMPCODE	contains one of the following constants: MQCC_OK MQCC_WARNING See note at the beginning of this table.
IMFMQI_CFH_REASON	contains a numeric reason code between 0 and 999999999
IMFMQI_CFH_PARAMETERCOUNT	contains decimal data

Appendix E Conversion Checklist

Converting to the Current Release of MAINVIEW AutoOPERATOR

In MAINVIEW AutoOPERATOR release 6.2, your 5.1 automation works exactly as it did in release 5.1. However, by converting your 5.1 automation, you can use the new features that are available in release 6.2. This checklist describes how you can change your 5.1 automation to use 6.2 functions.

Note: These changes are not required when migrating from MAINVIEW AutoOPERATOR 6.1 to MAINVIEW AutoOPERATOR 6.2.

Rule Processing

In MAINVIEW AutoOPERATOR 5.1, two types of MQSeries messages are recognized by the rule processor. They are MQEVENT and USER messages, specified under the Type field on the Selection Criteria panel. In MAINVIEW AutoOPERATOR 6.2, the Format field replaces the Type field used in release 5.1. Any known IBM format name constant can be specified.

The following list describes valid entries for the Format field:

MQFMT_EVENT	Event message
MQFMT_NONE	No format name
MQFMT_ADMIN	Command server request/reply message
MQFMT_CHANNEL_COMPLETED	Channel completed 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_IMS	IMS information header
MQFMT_IMS_VAR_STRING	IMS variable string
MQFMT_MD_EXTENSION	Message-descriptor extension
MQFMT_PCF	User-defined message in programmable command format
MQFMT_REF_MSG_HEADER	Reference message header
MQFMT_STRING	Message consisting entirely of characters
MQFMT_TRIGGER	Trigger message
MQFMT_WORK_INFO_HEADER	Work information header
MQFMT_XMIT_Q_HEADER	Transmission queue header
USER	Any message format except MQFMT_EVENT
RFH	Rules and Formatting Header
RFH2	Rules and Formatting Header – version 2

Note: To preserve compatibility, Rules created in 5.1 using MQEVENT and USER will continue to work and will not be changed by the MAINVIEW AutoOPERATOR 6.2 Rule Processor.

- MQSeries variables (IMFQxxx) used in Rules should be changed to use the corresponding new-style variable for MAINVIEW AutoOPERATOR 6.2, for example: IMFQ_MD_*. See Chapter 5, “Automating MQSeries Events”.
- New operators are available to use when comparing MQSeries data that is hexadecimal or binary. These operators are available on the MsgId, CorrelId and Msg Buffer fields of the Selection Criteria and the Variable Dependency panels. See Chapter 5, “Automating MQSeries Events”.
- The Selection Criteria panel has new Substring and length fields adjacent to the MsgId, CorrelId and Msg Buffer fields that you can use to target comparisons at specific locations within the respective field or buffer.

- The Action Specification panel has two new fields:
 - **Destination QMgr**—You can use this field to specify that a message being copied or moved should go to another queue manager.
 - **Remove DLH**—You can use this field to remove a message dead letter header and the original message that was rebuilt during a copy or move operation.
- Note:** A message passed to a Rule-invoked EXEC will not contain the dead letter header when using this option.
- The AAOMQLxx BBPARM member can be updated to specify that a queue should be opened as SHARE or EXCLUSIVE. See Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” for details.
 - The AAOMQLxx BBPARM member can be updated to specify that the Rule processor should see messages present on the queue when MAINVIEW AutoOPERATOR opens the queue for possible automation. See Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” for details.

EXEC Manager

The following information addresses the comparison between MAINVIEW AutoOPERATOR 5.1 and MAINVIEW AutoOPERATOR 6.2 EXEC managers:

- For every IMFEXEC MQ statement in MAINVIEW AutoOPERATOR 5.1, there is a corresponding IMFEXEC MQI statement in MAINVIEW AutoOPERATOR 6.2. To use the new statements, you must use the new variables. In addition, MAINVIEW AutoOPERATOR 6.2 uses the IBM constant names as variable values. See “Rule Processing” on page E-1 for details.
- Variables used for IMFEXEC MQI statements have a different format than those used with the IMFEXEC MQ statement from MAINVIEW AutoOPERATOR 5.1. The variables are made up of a common prefix (IMFMQL_), followed by the MQSeries structure ID (MD_, DLH_, etc.), followed by the field name from the MQSeries structure (Format, Expire, etc.). You must use these variables when converting to IMFEXEC MQI. See Chapter 2, “Implementing MAINVIEW AutoOPERATOR for MQSeries” for details.

- Options variables in MAINVIEW AutoOPERATOR 6.2 now use complete constant names instead of partial constant names as in MAINVIEW AutoOPERATOR 5.1. For example, options for an IMFEXEC MQI OPEN statement may be set as follows:
IMFMQI_OO_OPTIONS = 'MQOO_OUTPUT'. See Chapter 5, “Automating MQSeries Events”.
- The Options parameter that was available on MAINVIEW AutoOPERATOR 5.1 IMFEXEC MQ statements (OOPTS, COPTS, GOPTS and POPTS) is still available. However, the values specified in MAINVIEW AutoOPERATOR 6.2 consist of the full IBM constant name. For example, MQGMO_NO_SYNCPOINT instead of NO_SYNCPOINT, as specified in MAINVIEW AutoOPERATOR 5.1. See Chapter 5, “Automating MQSeries Events”.
- The IMFEXEC COPY MQ statement in MAINVIEW AutoOPERATOR 5.1 has been replaced in MAINVIEW AutoOPERATOR 6.2 with the IMFEXEC COPY MQI statement. Using this statement, you can specify which structures in the message should have the COPY of their variables performed. See Chapter 5, “Automating MQSeries Events”.
- The IMFEXEC DISPLAY MQ statement in MAINVIEW AutoOPERATOR 5.1 has been replaced in MAINVIEW AutoOPERATOR 6.2 with the IMFEXEC DISPLAY MQI statement. Using this statement, you can specify which structure variables should be written to the BBI journal. See Chapter 5, “Automating MQSeries Events”.

Index

A

AAOMQL00

- modifying settings 2-9
- parameters 3-5

AAOPRM00

- modifying settings 2-8
- parameters 3-1

Action Specification - MQS

- field description
 - Cmd(Type) 5-68
 - Destination QMgr 5-71
 - Destination Queue 5-70
 - DOM Id 5-70
 - EXEC Name/Parms 5-68
 - Info 5-70
 - Issue WTO Msg 5-70
 - Keep Message 5-70
 - Notify 5-69
 - Remove DLH 5-71
 - Set Variable 5-69
- panel 5-16, 5-17, 5-27, 5-28, 5-67, 5-68

ACTN 5-72

Alert Actions II - MQS panel 5-18, 5-29

Alert Actions MQS panel 5-17, 5-28

ALIAS_BASE_Q_TYPE_ERROR

- IMFQ_EVENT_APPLNAME 5-33
- IMFQ_EVENT_APPLTYPE 5-32
- IMFQ_EVENT_BASEQNAME 5-32
- IMFQ_EVENT_QMGRNAME 5-33
- IMFQ_EVENT_QNAME 5-32
- IMFQ_EVENT_QTYPE 5-32
- instrumentation event 5-32, 5-33

APF-Authorized libraries, adding 2-8

APL variables

- APLCC 7-4
- APLLN1 7-5
- APLLNxx 7-5
- APLNOL 7-5
- APLRC 7-4
- IMFRC 7-4

APLCC, APL variable 7-4

APLLN1, APL variable 7-5

APLLNxx, APL variable 7-5

APLNOL, APL variable 7-5

APLRC, APL variable 7-4

applications communicating with MQSeries objects 9-4

AUTHOREV

- NOT_AUTHORIZED TYPE 5-41
- queue manager attribute 1-7

Authority event type 1-7

automating events 5-1

automation

- problem examples A-19
- tracking MQS event statistics 5-71
- viewing statistics 4-1

Automation Control panel 5-8, 5-22, 5-23

Automation Power Line 7-1 to 7-7

- APLCC variables 7-4
- APLRC variables 7-4
- CMD parameter 7-2
- DEBUG parameter 7-2
- examples 7-6
- HELP parameter 7-3
- IMFRC variables 7-4
- NODE parameter 7-2
- parameters 7-2
- PORT parameter 7-2
- RM parameter 7-2
- WAIT parameter 7-2

AutoOPERATOR

- connecting to MQSeries 2-1
- interaction with MQSeries 1-8 to 1-10

AutoOPERATOR for MQSeries

- activating 2-6
- customizing 3-1
- description 1-1
- diagnosing errors A-1
- IMFEXEC statements, using 9-1
- implementing 2-1
- MQI EXEC interface, description 9-4
- not monitoring A-11
- obtaining status 2-33
- OCD 4-1
- securing 8-1
- software requirements 2-3
- solutions 6-1
- terms and concepts 1-2
- workstation panel 4-1

AutoOPERATOR-supplied variable
for all MQSeries messages 5-55, 5-56
IMFOQMGR 5-56
IMFQ_MD_ACCOUNTINGTOKEN 5-59
IMFQ_MD_APPLIDENTITYDATA 5-59
IMFQ_MD_APPLORIGINDATA 5-60
IMFQ_MD_BACKOUTCOUNT 5-59
IMFQ_MD_CODEDCHARSETID 5-58
IMFQ_MD_CORRELID 5-59
IMFQ_MD_ENCODING 5-58
IMFQ_MD_EXPIRY 5-57
IMFQ_MD_FEEDBACK 5-58
IMFQ_MD_FORMAT 5-59
IMFQ_MD_GROUPID 5-60
IMFQ_MD_MSGFLAGS 5-61
IMFQ_MD_MSGID 5-59
IMFQ_MD_MSGSEQNUMBER 5-60
IMFQ_MD_MSGTYPE 5-57
IMFQ_MD_OFFSET 5-60
IMFQ_MD_ORIGINALLENGTH 5-61
IMFQ_MD_PERSISTENCE 5-59
IMFQ_MD_PRIORITY 5-59
IMFQ_MD_PUTAPPLNAME 5-60
IMFQ_MD_PUTAPPLTYPE 5-60
IMFQ_MD_PUTDATE 5-60
IMFQ_MD_PUTTIME 5-60
IMFQ_MD_REPLYTOQ 5-59
IMFQ_MD_REPLYTOQMGR 5-59
IMFQ_MD_REPORT 5-57
IMFQ_MD_STRUCID 5-56
IMFQ_MD_USERIDENTIFIER 5-59
IMFQ_MD_VERSION 5-56
IMFQ_QAO_CATCH_UP_MSG 5-56
IMFQ_QATTR_BACKOUTREQUEUEQNA
ME 5-61
IMFQ_QATTR_BACKOUTTHRESHOLD
5-61
IMFQ_QATTR_CREATIONDATE 5-61
IMFQ_QATTR_CREATIONTIME 5-61
IMFQ_QATTR_CURRENTQDEPTH 5-61
IMFQ_QATTR_DEFINITIONTYPE 5-61
IMFQ_QATTR_DEFINPUTOPENOPTION
5-61
IMFQ_QATTR_DEFPERSISTENCE 5-61
IMFQ_QATTR_DEFPRIORITY 5-61
IMFQ_QATTR_HARDENGETBACKOUT
5-61
IMFQ_QATTR_INHIBITGET 5-61

AutoOPERATOR-supplied variable (*continued*)
IMFQ_QATTR_INHIBITPUT 5-62
IMFQ_QATTR_INITIATIONQNAME
5-61
IMFQ_QATTR_MAXMSGLENGTH 5-62
IMFQ_QATTR_MAXQDEPTH 5-62
IMFQ_QATTR_MSGDELIVERYSEQUEN
CE 5-62
IMFQ_QATTR_OPENINPUTCOUNT 5-62
IMFQ_QATTR_OPENOUTPUTCOUNT
5-62
IMFQ_QATTR_PROCESSNAME 5-62
IMFQ_QATTR_QDESC 5-61
IMFQ_QATTR_RETENTIONINTERVAL
5-62
IMFQ_QATTR_SHAREABILITY 5-62
IMFQ_QATTR_STORAGECLASS 5-62
IMFQ_QATTR_TRIGGERCONTROL 5-62
IMFQ_QATTR_TRIGGERDAT 5-62
IMFQ_QATTR_TRIGGERDEPTH 5-62
IMFQ_QATTR_TRIGGERMSGPRIORITY
5-62
IMFQ_QATTR_TRIGGERTYPE 5-62
IMFQ_QATTR_USAGE 5-62
IMFQ_STRUCTURES 5-56
IMFQCPF 5-56
IMFQRMT 5-56

B

BACK (MQI command) 9-13
basic intercommunication solution
description 6-24
implementing 6-27
MQCOR000 Rule 6-25
MQCOR001 Rule 6-25
MQCOR006 Rule 6-25
MQCOR007 Rule 6-25
MQCOR008 Rule 6-25
MQCOR009 Rule 6-26
MQCOR010 Rule 6-26
MQCOR011 Rule 6-26
MQCOR012 Rule 6-26
MQCOR013 Rule 6-26
MQCOR014 Rule 6-26
MQCOR015 Rule 6-26
MQCOR016 Rule 6-26

basic intercommunication solution (*continued*)

- MQCOR017 Rule 6-26
- MQCOR018 Rule 6-26
- MQCOR019 Rule 6-26
- MQCORMOD Rule 6-25
- QMQCORCF EXEC 6-25
- stopping 6-28

BBI Control Command 2-33

BBI-SS PAS, restart 2-13

BBOMVAO.EXEC.REPLY 9-60

BBPARAM

- AAOMQL00 parameters 3-5
- parameters for AAOPRM00 3-1

BRIDGE_STARTED 5-33

BRIDGE_STOPPED 5-34

BUFFER 9-38

BUFFLEN 9-39

C

CHADEV, queue manager attribute 1-8

channel availability solution

- implementing 6-21

- MQP00001 6-20

- MQP2023C 6-20

- MQP2023D 6-20

- MQP20636 6-20

- MQP20832 6-20

- MQP2083C 6-20

- MQP54500 6-20

- MQPRMT01 6-20

- MQTCK001 6-20

- MQTCK001 Rule 6-21

- Rule IDs 6-20

- stopping 6-23

channel event, description 1-7

CHANNEL_ACTIVATED 5-34

CHANNEL_CONV_ERROR 5-35

CHANNEL_NOT_ACTIVATED 5-36

CHANNEL_START event 6-20

CHANNEL_STARTED 5-36

CHANNEL_STOP event 6-20

CHANNEL_STOPPED

- IMFQ_EVENT_AUXERRORDATAINT1 5-37

- IMFQ_EVENT_AUXERRORDATAINT2 5-37

- IMFQ_EVENT_AUXERRORDATASTR1 5-37

- IMFQ_EVENT_AUXERRORDATASTR2 5-37

- IMFQ_EVENT_AUXERRORDATASTR3 5-37

- IMFQ_EVENT_CHANNELNAME 5-36

- IMFQ_EVENT_CONNECTIONNAME 5-36

- IMFQ_EVENT_ERRORIDENTIFIER 5-37

- IMFQ_EVENT_QMGRNAME 5-36

- IMFQ_EVENT_REASONQUALIFIER 5-37

- IMFQ_EVENT_XMITQNAME 5-36
- instrumentation event 5-36, 5-37

checklist, diagnostic A-4

CLOSE (MQI command), example 9-15

CMD parameter

- installation verification 2-28

- Power Line 7-2

CMD Type(MQS), examples 9-62

CMIT (MQI command), example 9-17

coding, sample B-1

Command MQ Automation Power Line 7-1 to 7-7

Command MQ On Ramp

- color diagnosis 2-20

- description 2-23

- diagnosing errors 2-22

- hyperlinking 2-16

- logging off 2-21

- menu options 2-17

- queue manager connection 2-20

- removing non-MVS queue managers 2-19

- specifying MVS queue manager 2-18

- specifying non-MVS queue manager 2-18

- user interface 2-17

commands, BBI control 2-33

communicating with MQSeries objects 9-4

completion code, returning 9-5

Confirm Rule Set Modifications panel 5-20, 5-31

confirming rule set modification, warning 5-20

CONN (MQI command), example 9-19
connecting to MQSeries 2-1
connection
 manual 2-23
 status color 2-20
 using the Command MQ On Ramp 2-20
connection handle 9-4
connectivity
 defining 2-14
 setting up 2-15
COPY MQI command), example 9-63
CQ parameter, CMD 9-59
CSQ90221 message 9-60

D

.D ACTIVE 2-33
.DISPLAY ACTIVE 2-33
.D KEYS 2-33
.DISPLAY KEYS 2-33
.D PRODUCTS 2-33
.DISPLAY PRODUCTS 2-33
Dead Letter Aging Management, enabling Rule
 6-6
Dead Letter Management
 enabling Rules 6-4
 invalid dead letters 6-4
 valid dead letters 6-5
dead letter queue 1-4
Dead Letter Queue solution
 Dead Letter Management 6-4
 implementing 6-3
 stopping 6-7
 using 6-2
DEBUG, Power Line parameter 7-2
DEF_XMIT_Q_TYPE_ERROR 5-38
DEF_XMIT_Q_USAGE_ERROR 5-39
Define Queue for Non-OS/390 Instrumentation
 Events 2-13
defining connectivity 2-14
diagnosing errors
 AutoOPERATOR for MQSeries A-1
 Command MQ On Ramp 2-22
 tools A-2
diagnostic checklist A-4
diagnostic rules, enabling 2-11

DISC (MQI command), example 9-21
DISPLAY MQI 9-1, 9-66

E

error messages, resources A-1
event message, description 5-3
events
 automating 5-1
 automation 1-2
 channel 1-7
 description 1-5
 instrumentation 5-32 to 5-54
 listening 2-24
 not firing A-14
 performance 1-7
 queue manager 1-7
 tracking automation statistics 5-71
 types 1-7
 variables 5-55
EXEC timing out, non-MVS queue manager
 A-24

F

Formatting Header Variables C-23, D-28
Formatting Header Variables Version 2 C-24,
 D-29
frequently asked questions A-6

G

generating
 instrumentation events 3-3
 MQEVENTs 3-3
GET (MQI command), example 9-23
Get(Disabled), queues set to 3-3
GET_INHIBITED 5-40

H

HCONN 9-33, 9-37
HELP, Power Line parameter 7-3
HOBJ parameters, HOBJ 9-37
hyperlinking, Command MQ On Ramp 2-16

IE, installation verification 2-28

IMFEXEC CMD

CQ parameter 9-59

examples 9-62

LM parameter 9-58

MQ command parameter 9-58

PCF parameter 9-59

QM parameter 9-59

Response parameter 9-59

Type parameter 9-58

Wait parameter 9-59

IMFEXEC MQI commands (description of),

COPY MQI 9-1

IMFEXEC MQI PUT1 parameters, HCONN
9-47

IMFEXEC MQI variables 9-2

IMFMQI 9-30

IMFMQI_MD_ACCOUNTINGTOKEN 9-29,
9-42, 9-53

IMFMQI_MD_APPLIDENTITYDATA 9-29,
9-42, 9-53

IMFMQI_MD_APPLORIGINDATA 9-30, 9-45

IMFMQI_MD_BACKOUTCOUNT 9-29, 9-42,
9-53

IMFMQI_MD_CODEDCHARSETID 9-28,
9-41, 9-52

IMFMQI_MD_CORRELID 9-29, 9-42, 9-53

IMFMQI_MD_ENCODING 9-28, 9-41, 9-52

IMFMQI_MD_EXPIRY 9-27, 9-40, 9-51

IMFMQI_MD_FEEDBACK 9-27, 9-41, 9-52

IMFMQI_MD_FORMAT 9-28, 9-41, 9-52

IMFMQI_MD_GROUPID 9-30, 9-45

IMFMQI_MD_MSGFLAGS 9-31, 9-45

IMFMQI_MD_MSGID 9-28, 9-42, 9-53

IMFMQI_MD_MSGSEQNUMBER 9-30, 9-45

IMFMQI_MD_MSGTYPE 9-27, 9-40, 9-51

IMFMQI_MD_OFFSET 9-30, 9-45

IMFMQI_MD_ORIGINALLENGTH 9-31, 9-45

IMFMQI_MD_PERSISTENCE 9-28, 9-42, 9-53

IMFMQI_MD_PRIORITY 9-28, 9-41, 9-53

IMFMQI_MD_PUTAPPLNAME 9-30, 9-44

IMFMQI_MD_PUTAPPLTYPE 9-30, 9-44

IMFMQI_MD_PUTDATE 9-30, 9-44

IMFMQI_MD_PUTTIME 9-30, 9-44

IMFMQI_MD_REPLYTOQ 9-29, 9-42, 9-53

IMFMQI_MD_REPLYTOQMGR 9-29, 9-42,
9-53

IMFMQI_MD_REPORT 9-27, 9-40, 9-51

IMFMQI_MD_STRUCID 9-26, 9-40, 9-50

IMFMQI_MD_USERIDENTIFIER 9-29, 9-42,
9-53

IMFMQI_MD_VERSION 9-26, 9-40, 9-50

IMFMQI_OD_ALTERNATESECURITYID
9-35, 9-50

IMFMQI_OD_ALTERNATEUSERID 9-35,
9-50

IMFMQI_OD_DYNAMICQNAME 9-35, 9-50

IMFMQI_OD_INVALIDDESTCOUNT 9-35,
9-50

IMFMQI_OD_KNOWNDESTCOUNT 9-35,
9-50

IMFMQI_OD_OBJECTNAME 9-34, 9-49

IMFMQI_OD_OBJECTQMGRNAME 9-35,
9-49

IMFMQI_OD_OBJECTRECOFFSET 9-35,
9-50

IMFMQI_OD_OBJECTRECPTR 9-35, 9-50

IMFMQI_OD_OBJECTTYPE 9-34, 9-49

IMFMQI_OD_RECSPRESENT 9-35, 9-50

IMFMQI_OD_RESOLVEDQMGRNAME
9-35, 9-50

IMFMQI_OD_RESOLVEDQNAME 9-35, 9-50

IMFMQI_OD_RESPONSERECOFFSET 9-35,
9-50

IMFMQI_OD_RESPONSERECPTR 9-35, 9-50

IMFMQI_OD_STRUCID 9-34, 9-49

IMFMQI_OD_UNKNOWNDESTCOUNT
9-35, 9-50

IMFMQI_OD_VERSION 9-34, 9-49

IMFMQI_STRUCTURES 9-26

IMFOQMGR 5-56

IMFQ_EVENT_APPLNAME 5-50

ALIAS_BASE_Q_TYPE_ERROR 5-33

DEF_XMIT_Q_TYPE_ERROR 5-38

DEF_XMIT_Q_USAGE_ERROR 5-39

GET_INHIBITED 5-40

NOT_AUTHORIZED 5-41

PUT_INHIBITED 5-42

Q_TYPE_ERROR 5-46

REMOTE_Q_NAME_ERROR 5-47

IMFQ_EVENT_APPLNAME (*continued*)
 UNKNOWN_ALIAS_BASE_Q 5-48
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_REMOTE_Q_MGR 5-51
 UNKNOWN_XMIT_Q 5-52
 XMIT_Q_USAGE_ERROR 5-54

IMFQ_EVENT_APPLTYPE
 ALIAS_BASE_Q_TYPE_ERROR 5-32
 DEF_XMIT_Q_TYPE_ERROR 5-38
 DEF_XMIT_Q_USAGE_ERROR 5-39
 GET_INHIBITED 5-40
 NOT_AUTHORIZED 5-41
 PUT_INHIBITED 5-42
 Q_TYPE_ERROR 5-46
 REMOTE_Q_NAME_ERROR 5-47
 UNKNOWN_ALIAS_BASE_Q 5-48
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_OBJECT_NAME 5-50
 UNKNOWN_REMOTE_Q_MGR 5-51
 UNKNOWN_XMIT_Q 5-52
 XMIT_Q_TYPE_ERROR 5-53
 XMIT_Q_USAGE_ERROR 5-54

IMFQ_EVENT_AUXERRORDATAINT1,
 CHANNEL_STOPPED 5-37

IMFQ_EVENT_AUXERRORDATAINT2,
 CHANNEL_STOPPED 5-37

IMFQ_EVENT_AUXERRORDATASTR1,
 CHANNEL_STOPPED 5-37

IMFQ_EVENT_AUXERRORDATASTR2,
 CHANNEL_STOPPED 5-37

IMFQ_EVENT_AUXERRORDATASTR3,
 CHANNEL_STOPPED 5-37

IMFQ_EVENT_BASEQNAME
 ALIAS_BASE_Q_TYPE_ERROR 5-32
 UNKNOWN_ALIAS_BASE_Q 5-48

IMFQ_EVENT_BRIDGENAME
 BRIDGE_STARTED 5-33
 BRIDGE_STOPPED 5-34

IMFQ_EVENT_BRIDGETYPE
 BRIDGE_STARTED 5-33
 BRIDGE_STOPPED 5-34

IMFQ_EVENT_CHANNELNAME
 CHANNEL_ACTIVATED 5-34
 CHANNEL_CONV_ERROR 5-35
 CHANNEL_NOT_ACTIVATED 5-36
 CHANNEL_STARTED 5-36
 CHANNEL_STOPPED 5-36

IMFQ_EVENT_COMMAND,
 NOT_AUTHORIZED 5-41

IMFQ_EVENT_CONNECTIONNAME
 CHANNEL_ACTIVATED 5-34
 CHANNEL_CONV_ERROR 5-35
 CHANNEL_NOT_ACTIVATED 5-36
 CHANNEL_STARTED 5-36
 CHANNEL_STOPPED 5-36

IMFQ_EVENT_ERRORIDENTIFIER
 BRIDGE_STOPPED 5-34
 CHANNEL_STOPPED 5-37

IMFQ_EVENT_FORMAT,
 CHANNEL_CONV_ERROR 5-35

IMFQ_EVENT_HIGHQDEPTH
 Q_DEPTH_HIGH 5-43
 Q_DEPTH_LOW 5-43
 Q_FULL 5-44
 Q_SERVICE_INTERVAL_HIGH 5-45

IMFQ_EVENT_MSGDEQCOUNT
 Q_DEPTH_HIGH 5-43
 Q_DEPTH_LOW 5-43
 Q_SERVICE_INTERVAL_HIGH 5-45

IMFQ_EVENT_MSGENQCOUNT
 Q_DEPTH_HIGH 5-43
 Q_DEPTH_LOW 5-43
 Q_FULL 5-44
 Q_SERVICE_INTERVAL_HIGH 5-45

IMFQ_EVENT_OBJECTQMGRNAME
 DEF_XMIT_Q_TYPE_ERROR 5-38
 DEF_XMIT_Q_USAGE_ERROR 5-39
 NOT_AUTHORIZED 5-42
 PUT_INHIBITED 5-42
 Q_TYPE_ERROR 5-46
 REMOTE_Q_NAME_ERROR 5-47
 UNKNOWN_ALIAS_BASE_Q 5-48
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_OBJECT_NAME 5-50
 UNKNOWN_REMOTE_Q_MGR 5-51
 UNKNOWN_XMIT_Q 5-52
 XMIT_Q_TYPE_ERROR 5-53
 XMIT_Q_USAGE_ERROR 5-54

IMFQ_EVENT_OPTIONS,
 NOT_AUTHORIZED 5-41

IMFQ_EVENT_PROCESSNAME
 NOT_AUTHORIZED 5-42
 UNKNOWN_OBJECT_NAME 5-50

IMFQ_EVENT_QMGRNAME
 ALIAS_BASE_Q_TYPE_ERROR 5-33
 BRIDGE_STARTED 5-33
 BRIDGE_STOPPED 5-34
 CHANNEL_ACTIVATED 5-34
 CHANNEL_CONV_ERROR 5-35
 CHANNEL_NOT_ACTIVATED 5-36
 CHANNEL_STARTED 5-36
 CHANNEL_STOPPED 5-36
 DEF_XMIT_Q_TYPE_ERROR 5-38
 DEF_XMIT_Q_USAGE_ERROR 5-39
 GET_INHIBITED 5-40
 NOT_AUTHORIZED 5-41
 PUT_INHIBITED 5-42
 Q_DEPTH_HIGH 5-43
 Q_DEPTH_LOW 5-43
 Q_FULL 5-44
 Q_MGR_ACTIVE 5-44
 Q_MGR_NOT_ACTIVE 5-44
 Q_SERVICE_INTERVAL_HIGH 5-45
 Q_TYPE_ERROR 5-46
 REMOTE_Q_NAME_ERROR 5-47
 UNKNOWN_ALIAS_BASE_Q 5-48
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_OBJECT_NAME 5-50
 UNKNOWN_REMOTE_Q_MGR 5-51
 UNKNOWN_XMIT_Q 5-52
 XMIT_Q_TYPE_ERROR 5-53
 XMIT_Q_USAGE_ERROR 5-54

IMFQ_EVENT_QNAME
 ALIAS_BASE_Q_TYPE_ERROR 5-32
 DEF_XMIT_Q_TYPE_ERROR 5-38
 DEF_XMIT_Q_USAGE_ERROR 5-39
 GET_INHIBITED 5-40
 NOT_AUTHORIZED 5-41
 PUT_INHIBITED 5-42
 Q_DEPTH_HIGH 5-43
 Q_DEPTH_LOW 5-43
 Q_FULL 5-44
 Q_SERVICE_INTERVAL_HIGH 5-45
 Q_TYPE_ERROR 5-46
 REMOTE_Q_NAME_ERROR 5-47
 UNKNOWN_ALIAS_BASE_Q 5-48
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_OBJECT_NAME 5-50
 UNKNOWN_REMOTE_Q_MGR 5-51
 UNKNOWN_XMIT_Q 5-52

IMFQ_EVENT_QNAME (*continued*)
 XMIT_Q_TYPE_ERROR 5-53
 XMIT_Q_USAGE_ERROR 5-54

IMFQ_EVENT_QTYPE
 ALIAS_BASE_Q_TYPE_ERROR 5-32
 DEF_XMIT_Q_TYPE_ERROR 5-38
 XMIT_Q_TYPE_ERROR 5-53

IMFQ_EVENT_REASONQUALIFIER
 BRIDGE_STOPPED 5-34
 CHANNEL_CONV_ERROR 5-35
 CHANNEL_STOPPED 5-37
 NOT_AUTHORIZED 5-41
 Q_MGR_NOT_ACTIVE 5-44

IMFQ_EVENT_TIMESINCERESET
 Q_DEPTH_HIGH 5-43
 Q_DEPTH_LOW 5-43
 Q_FULL 5-44
 Q_SERVICE_INTERVAL_HIGH 5-45

IMFQ_EVENT_USERIDENTIFIER,
 NOT_AUTHORIZED 5-41

IMFQ_EVENT_XMITQNAME
 CHANNEL_ACTIVATED 5-34
 CHANNEL_CONV_ERROR 5-35
 CHANNEL_NOT_ACTIVATED 5-36
 CHANNEL_STARTED 5-36
 CHANNEL_STOPPED 5-36
 DEF_XMIT_Q_TYPE_ERROR 5-38
 DEF_XMIT_Q_USAGE_ERROR 5-39
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_XMIT_Q 5-52
 XMIT_Q_TYPE_ERROR 5-53
 XMIT_Q_USAGE_ERROR 5-54

IMFQ_MD_ACCOUNTINGTOKEN 5-59
 IMFQ_MD_APPLIDENTITYDATA 5-59
 IMFQ_MD_APPLORIGINDATA 5-60
 IMFQ_MD_BACKOUTCOUNT 5-59
 IMFQ_MD_CODEDCHARSETID 5-58
 IMFQ_MD_CORRELID 5-59
 IMFQ_MD_ENCODING 5-58
 IMFQ_MD_EXPIRY 5-57
 IMFQ_MD_FEEDBACK 5-58
 IMFQ_MD_FORMAT 5-59
 IMFQ_MD_GROUPID 5-60
 IMFQ_MD_MSGFLAGS 5-61
 IMFQ_MD_MSGID 5-59
 IMFQ_MD_MSGSEQNUMBER 5-60
 IMFQ_MD_MSGTYPE 5-57

IMFQ_MD_OFFSET 5-60
IMFQ_MD_ORIGINALLENGTH 5-61
IMFQ_MD_PERSISTENCE 5-59
IMFQ_MD_PRIORITY 5-59
IMFQ_MD_PUTAPPLNAME 5-60
IMFQ_MD_PUTAPPLTYPE 5-60
IMFQ_MD_PUTDATE 5-60
IMFQ_MD_PUTTIME 5-60
IMFQ_MD_REPLYTOQ 5-59
IMFQ_MD_REPLYTOQMGR 5-59
IMFQ_MD_REPORT 5-57
IMFQ_MD_STRUCID 5-56
IMFQ_MD_USERIDENTIFIER 5-59
IMFQ_MD_VERSION 5-56
IMFQ_QAO_CATCH_UP_MSG 5-56
IMFQ_QATTR_BACKOUTREQUEUEQNAME
E 5-61
IMFQ_QATTR_BACKOUTTHRESHOLD
5-61
IMFQ_QATTR_CREATIONDATE 5-61
IMFQ_QATTR_CREATIONTIME 5-61
IMFQ_QATTR_CURRENTQDEPTH 5-61
IMFQ_QATTR_DEFINITIONTYPE 5-61
IMFQ_QATTR_DEFINPUTOPENOPTION
5-61
IMFQ_QATTR_DEFPERSISTENCE 5-61
IMFQ_QATTR_DEFPRIORITY 5-61
IMFQ_QATTR_HARDENGETBACKOUT
5-61
IMFQ_QATTR_INHIBITGET 5-61
IMFQ_QATTR_INHIBITPUT 5-62
IMFQ_QATTR_INITIATIONQNAME 5-61
IMFQ_QATTR_MAXMSGLENGTH 5-62
IMFQ_QATTR_MAXQDEPTH 5-62
IMFQ_QATTR_MSGDELIVERYSEQUENCE
5-62
IMFQ_QATTR_OPENINPUTCOUNT 5-62
IMFQ_QATTR_OPENOUTPUTCOUNT 5-62
IMFQ_QATTR_PROCESSNAME 5-62
IMFQ_QATTR_QDESC 5-61
IMFQ_QATTR_RETENTIONINTERVAL 5-62
IMFQ_QATTR_SHAREABILITY 5-62
IMFQ_QATTR_STORAGECLASS 5-62
IMFQ_QATTR_TRIGGERCONTROL 5-62
IMFQ_QATTR_TRIGGERDAT 5-62
IMFQ_QATTR_TRIGGERDEPTH 5-62
IMFQ_QATTR_TRIGGERMSGPRIORITY
5-62

IMFQ_QATTR_TRIGGERTYPE 5-62
IMFQ_QATTR_USAGE 5-62
IMFQ_STRUCTURES 5-56
IMFQAPLN
 PUT_INHIBITED 5-33, 5-38 to 5-42, 5-46
 to 5-54
 XMIT_Q_TYPE_ERROR 5-53
IMFQCPF 5-56
IMFQRMT 5-56
IMFRC, APL variable 7-4
IMFTEXT, message size limitation 5-21
Inhibit, event type 1-7
INHIBTEV
 GET_INHIBITED 5-40
 PUT_INHIBITED 5-42
 queue manager attribute 1-7
installation troubleshooting A-6
installation verification procedure
 description 2-27
 examples 2-28
 required values 2-27
instrumentation events
 categories 1-7
 description 5-3, 5-32 to 5-54
 enabling
 MVS 2-13
 non-MVS 2-13
 generating 3-3
 types 1-7, 2-14
interface, MQSeries EXEC 9-2
invalid dead letters, Dead Letter Management
 6-4
issuing commands, set up 2-26

K

keys, specifying 2-6

L

libraries, adding 2-8
listening for events, set up 2-24
LM parameter, CMD 9-58
local queue manager 1-3
Local, event type 1-7

LOCALEV

- queue manager attribute 1-7
- UNKNOWN_ALIAS_BASE_Q 5-48
- UNKNOWN_OBJECT_NAME 5-50

logging off, Command MQ On Ramp 2-21

M

message

- CSQ9022I 9-60
- description 1-8
- format 1-8

message channel 1-4

Message Queue Interface (MQI) 1-3

messaging 1-2

migration considerations 2-4

MQ command parameter, CMD 9-58

MQ PUT, non-zero code return A-17

MQCOR000 Rule 6-25

MQCOR001 Rule 6-25

MQCOR006 Rule 6-25

MQCOR007 Rule 6-25

MQCOR008 Rule 6-25

MQCOR009 Rule 6-26

MQCOR010 Rule 6-26

MQCOR011 Rule 6-26

MQCOR012 Rule 6-26

MQCOR013 Rule 6-26

MQCOR014 Rule 6-26

MQCOR015 Rule 6-26

MQCOR016 Rule 6-26

MQCOR017 Rule 6-26

MQCOR018 Rule 6-26

MQCOR019 Rule 6-26

MQCORMOD Rule 6-25

MQDEDQ01 Rule, enabling 6-4

MQDEDQ02 Rule, enabling 6-6

MQDEDQ03 Rule, enabling 6-6

MQDIA001, enabling 2-11

MQDIA002, enabling 2-11

MQDIA003, enabling 2-11

MQEV, parameters 3-2

MQEVENT

- creating Rules 5-8
- description 5-3
- generating 3-3

MQGINHIB

- parameters 3-2
- using 3-4

MQI BACK 9-1

MQI CLOSE 9-1

MQI CMIT 9-1

MQI commands

- description, COPY MQI 9-1
- list 9-2

MQI CONN 9-1

MQI DISC 9-1

MQI EXEC interface, description 9-4

MQI GET 9-1

MQI OPEN 9-1

MQI PUT 9-1

MQI PUT1 9-1

MQI sample coding B-1

MQIEXEC Interface 9-4

MQINT001 Rule, enabling 6-16

MQINT002 Rule, enabling 6-16

mqmd_UserIdentifier 9-61

MQNSHARE

- parameters 3-2
- using 3-4

MQP00001 6-20

MQP2023C 6-20

MQP2023D 6-20

MQP20636 6-20

MQP20832 6-20

MQP2083C 6-20

MQP54500 6-20

MQPRMT01 Rule 6-20

MQQDP001, enabling 6-18

MQQDP002, enabling 6-18

MQS events description 5-2

MQS Selection Criteria 5-63

MQSeries

- advantages 1-3
- description 1-1
- interaction with AutoOPERATOR 1-8 to 1-10
- messages, format 5-2
- terms and concepts 1-2
- workstation 4-1

MQSeries Event Types panel 5-13, 5-14

MQSeries events
 automation 1-2, 5-1
 description 1-5
 enabling MQSeries events 5-5
 tracking automation statistics 5-71
 variables 5-55

MQSeries EXEC interface 9-2

MQSeries Instrumentation, events 5-32

MQSeries Objects 9-4

MQSeries OCD, queues managed 4-7

MQTCK001 6-20

N

NCMD, installation verification 2-28

NIE, installation verification 2-28

NODE, Power Line parameter 7-2

Non-Event Messages 5-4

Non-MQFMT_EVENT Message Rule
 creating 5-21
 MQSeries Message, example 5-22

non-MVS queue manager
 removing from Command MQ On Ramp
 2-19
 specifying in Command MQ On Ramp 2-18

NOSHARE, queues set as 3-4

NOT_AUTHORIZED
 IMFQ_EVENT_APPLNAME 5-41
 IMFQ_EVENT_APPLTYPE 5-41
 IMFQ_EVENT_COMMAND 5-41
 IMFQ_EVENT_OBJECTQMGRNAME
 5-42
 IMFQ_EVENT_OPTIONS 5-41
 IMFQ_EVENT_PROCESSNAME 5-42
 IMFQ_EVENT_QMGRNAME 5-41
 IMFQ_EVENT_QNAME 5-41
 IMFQ_EVENT_REASONQUALIFIER
 5-41
 IMFQ_EVENT_USERIDENTIFIER 5-41
 instrumentation event 5-41

O

object handle 9-4

OPEN (MQI command), example 9-33

P

panel
 Action Specification - MQS 5-16, 5-17,
 5-27, 5-28, 5-67
 Alert Actions 5-17, 5-28
 Alert Actions II 5-18, 5-29
 Automation Control 5-8, 5-22, 5-23
 Confirm Rule Set Modifications 5-20, 5-31
 MQSeries Event Types 5-13, 5-14
 Rule Processor Detail Control 5-10, 5-11,
 5-18, 5-24, 5-25, 5-29
 Rule Set Overview 5-9, 5-23, 5-30
 Selection Criteria - MQS 5-11, 5-14, 5-25,
 5-26, 5-63
 Variable Dependencies - MQS 5-15, 5-27

parameters
 BUFFER 9-38
 BUFFLEN 9-39
 HCONN 9-33, 9-37

password keys, specifying 2-6

PCF parameter, CMD 9-59

PERFMDEV
 Q_DEPTH_LOW 5-43
 Q_DEPTHI_HIGH 5-43
 Q_FULL 5-44
 Q_SERVICE_INTERVAL_HIGH 5-45
 queue manager attribute 1-7

performance events, description 1-7

performance statistics 4-5

PFC variables C-25

POPTS 9-38

PORT, Power Line parameter 7-2

Power Line
 APLCC variables 7-4
 APLRC variables 7-4
 CMD parameter 7-2
 DEBUG parameter 7-2
 description 7-1
 examples 7-6
 HELP parameter 7-3
 IMFRC variables 7-4
 NODE parameter 7-2
 PORT parameter 7-2
 RM parameter 7-2
 WAIT parameter 7-2

primary commands, workstation panel 4-3

PUT (MQI command), example 9-37
PUT_INHIBITED
 IMFQ_EVENT_APPLNAME 5-42
 IMFQ_EVENT_APPLTYPE 5-42
 IMFQ_EVENT_OBJECTQMGRNAME
 5-42
 IMFQ_EVENT_QMGRNAME 5-42
 IMFQ_EVENT_QNAME 5-42
 IMFQAPLN 5-33, 5-38, 5-39, 5-40, 5-41,
 5-42, 5-46, 5-47, 5-48, 5-49, 5-50, 5-51,
 5-52, 5-53, 5-54
 instrumentation event 5-42
PUT1 (MQI command), example 9-47

Q

Q_DEPTH_HIGH 5-43
Q_DEPTH_LOW 5-43
Q_FULL 5-44
Q_MGR_ACTIVE 5-44
Q_MGR_NOT_ACTIVE 5-44
Q_SERVICE_INTERVAL_HIGH 5-45
Q_TYPE_ERROR 5-46
QM parameter, CMD 9-59
QMQRORCF EXEC 6-25
QMQRDEDQ1 EXEC, invoking 6-4
QMQRNB001, sample coding B-1
QMQRNB002, sample coding B-1
QMQRNB003, sample coding B-2
QMQRNB004, sample coding B-2
QMQRNB005, sample coding B-2
QMQRNB006, sample coding B-2
QMQRNB007, sample coding B-2
QMQRNB008, sample coding B-2
QMQRNB009, sample coding B-2
QMQRUNLDQ 6-2
queue depth management solution
 enabling Rules 6-18
 implementing 6-17
 stopping 6-19
queue management, viewing workstation panel
 4-7

queue manager
 communicating 1-4
 connecting manually 2-23
 connecting with Command MQ On Ramp
 2-20
 description 1-3
 events 1-7
 local 1-3
 not connecting A-8
 remote 1-3
 setting up connectivity 2-15
 specifying in Command MQ On Ramp 2-18
queue manager attribute
 AUTHOREV 1-7
 CHADEV 1-8
 INHIBTEV 1-7
 LOCALEV 1-7
 PERFMEEV 1-7
 REMOTEEV 1-7
 STRSTPEV 1-7
queues
 dead letter 1-4
 specifying in AAOMQL00 3-5
 specifying in AAOMQL00, examples 3-7
queuing 1-1, 1-2

R

reason code, returning 9-5
refresh, list of queues 2-11
remote queue manager 1-3
Remote, event type 1-7
REMOTE_Q_NAME_ERROR 5-47
REMOTEEV
 DEF_XMIT_Q_TYPE_ERROR 5-38
 DEF_XMIT_Q_USAGE_ERROR 5-39
 Q_TYPE_ERROR 5-46
 queue manager attribute 1-7
 REMOTE_Q_NAME_ERROR 5-47
 UNKNOWN_DEF_XMIT_Q 5-49
 UNKNOWN_REMOTE_Q_MGR 5-51
 UNKNOWN_XMIT_Q 5-52
 XMIT_Q_TYPE_ERROR 5-53
 XMIT_Q_USAGE_ERROR 5-54

- .RESET MQ 2-33
 - description 2-33
 - refresh list of queues 2-11
- .RESET QM 2-33
- RESPONSE parameter, CMD 9-59
- RM, Power Line parameter 7-2
- Rule Processor Detail Control panel 5-10, 5-11, 5-18, 5-24, 5-25, 5-29, 5-30
- Rule Set Modifications Panel 5-20, 5-31
- Rule Set Overview Panel 5-23
- Rule Set Overview panel 5-9, 5-19, 5-23, 5-30
- Rule-invoked EXECs C-23, C-24, D-28, D-29
- Rules C-23, C-24, D-28, D-29
 - creating (examples) 5-8
 - description 5-2
 - event messages 5-7
 - how to save 5-31
 - MQFMT_EVENT message 5-7
 - MQMFT_EVENT example 5-8
 - non-event messages 5-7
 - Non-MQFMT_EVENT Message 5-21
 - not enabling non-MVS queue A-27

S

- sample coding B-1
- security
 - AutoOPERATOR for MQSeries 8-1
 - defining SSIDs 2-12
 - defining user IDs 2-12
- Selection Criteria - MQS
 - field description
 - CorrelId 5-66
 - Event Type 5-65
 - Format 5-64
 - Manager(s) 5-64
 - Msg Buffer 5-67
 - Msgid 5-65
 - Queue ID 5-64
 - panel 5-11, 5-14, 5-25, 5-26, 5-63
- service interval performance solution
 - enabling Rules 6-16
 - implementing 6-15
 - stopping 6-17

- Setting Up Non-OS/390 Queue Manager
 - Connectivity 2-23
- Setting Up OS/390 Queue Manager
 - Connectivity 2-23
- SHARED variables 5-55
- software requirements 2-3
- solution
 - basic intercommunication
 - implementing 6-27
 - stopping 6-28
 - using 6-24
 - channel availability
 - description 6-20
 - implementing 6-21
 - stopping 6-23
 - Dead Letter Queue
 - Dead Letter Management 6-4
 - description 6-2
 - implementing 6-3
 - not moving messages A-19
 - stopping 6-7
 - implementation considerations 6-1
 - queue depth management
 - enabling Rules 6-18
 - implementing 6-17
 - stopping 6-19
 - service interval performance
 - enabling Rules 6-16
 - implementing 6-15
 - stopping 6-17
 - system queue archival
 - enabling Rules 6-10
 - implementing 6-8
 - stopping 6-14
- Start and Stop, event type 1-7
- statistics, tracking automation 5-71
- STEPLIB DD Card, adding libraries 2-8
- STRSTPEV
 - Q_MGR_ACTIVE 5-44
 - Q_MGR_NOT_ACTIVE 5-44
 - queue manager attribute 1-7
- system queue archival solution
 - description 6-2
 - enabling Rules 6-10
 - implementing 6-8
 - stopping 6-14

T

tools, diagnosing errors A-2
tracking automation statistics 5-71
transmission queue 1-4
TYPE parameter, CMD 9-58

U

UNKNOWN_ALIAS_BASE_Q 5-48
UNKNOWN_DEF_XMIT_Q 5-49
UNKNOWN_OBJECT_NAME 5-50
UNKNOWN_REMOTE_Q_MGR 5-51
UNKNOWN_XMIT_Q 5-52

V

valid dead letter, Dead Letter Management 6-5
Variable Dependencies panel 5-15, 5-27
variables

APL

APLCC 7-4
APLLN1 7-5
APLLNxx 7-5
APLNOL 7-5
APLRC 7-4
IMFRC 7-4

AutoOPERATOR-supplied

for USER messages 5-56
IMFOQMGR 5-56
IMFQ_MD_ACCOUNTINGTOKEN
5-59
IMFQ_MD_APPLIDENTITYDATA
5-59
IMFQ_MD_APPLORIGINDATA 5-60
IMFQ_MD_BACKOUTCOUNT 5-59
IMFQ_MD_CODEDCHARSETID 5-58
IMFQ_MD_CORRELID 5-59
IMFQ_MD_ENCODING 5-58
IMFQ_MD_EXPIRY 5-57
IMFQ_MD_FEEDBACK 5-58
IMFQ_MD_FORMAT 5-59
IMFQ_MD_GROUPID 5-60
IMFQ_MD_MSGFLAGS 5-61
IMFQ_MD_MSGID 5-59
IMFQ_MD_MSGSEQNUMBER 5-60

variables, AutoOPERATOR-supplied
(continued)

IMFQ_MD_MSGTYPE 5-57
IMFQ_MD_OFFSET 5-60
IMFQ_MD_ORIGINALLENGTH 5-61
IMFQ_MD_PERSISTENCE 5-59
IMFQ_MD_PRIORITY 5-59
IMFQ_MD_PUTAPPLNAME 5-60
IMFQ_MD_PUTAPPLTYPE 5-60
IMFQ_MD_PUTDATE 5-60
IMFQ_MD_PUTTIME 5-60
IMFQ_MD_REPLYTOQ 5-59
IMFQ_MD_REPLYTOQMGR 5-59
IMFQ_MD_REPORT 5-57
IMFQ_MD_STRUCID 5-56
IMFQ_MD_USERIDENTIFIER 5-59
IMFQ_MD_VERSION 5-56
IMFQ_QAO_CATCH_UP_MSG 5-56
IMFQ_QATTR_BACKOUTREQUEUE
QNAME 5-61
IMFQ_QATTR_BACKOUTTHRESHO
LD 5-61
IMFQ_QATTR_CREATIONDATE
5-61
IMFQ_QATTR_CREATIONTIME
5-61
IMFQ_QATTR_CURRENTQDEPTH
5-61
IMFQ_QATTR_DEFINITIONTYPE
5-61
IMFQ_QATTR_DEFINPUTOPENOPT
ION 5-61
IMFQ_QATTR_DEFPERSISTENCE
5-61
IMFQ_QATTR_DEFPRIORITY 5-61
IMFQ_QATTR_HARDENGETBACK
OUT 5-61
IMFQ_QATTR_INHIBITGET 5-61
IMFQ_QATTR_INHIBITPUT 5-62
IMFQ_QATTR_INITIATIONQNAME
5-61
IMFQ_QATTR_MAXMSGLENGTH
5-62
IMFQ_QATTR_MAXQDEPTH 5-62
IMFQ_QATTR_MSGDELIVERYSEQ
UENCE 5-62

variables, AutoOPERATOR-supplied

(continued)

IMFQ_QATTR_OPENINPUTCOUNT
5-62

IMFQ_QATTR_OPENOUTPUTCOUN
T 5-62

IMFQ_QATTR_PROCESSNAME 5-62

IMFQ_QATTR_QDESC 5-61

IMFQ_QATTR_RETENTIONINTERV
AL 5-62

IMFQ_QATTR_SHAREABILITY 5-62

IMFQ_QATTR_STORAGECLASS
5-62

IMFQ_QATTR_TRIGGERCONTROL
5-62

IMFQ_QATTR_TRIGGERDAT 5-62

IMFQ_QATTR_TRIGGERDEPTH
5-62

IMFQ_QATTR_TRIGGERMSGPRIOR
ITY 5-62

IMFQ_QATTR_TRIGGERTYPE 5-62

IMFQ_QATTR_USAGE 5-62

IMFQ_STRUCTURES 5-56

IMFQCPF 5-56

IMFQRMT 5-56

CICS Information Header C-6, D-11

Dead Letter Header C-3, D-8

IMFEXEC MQI 9-2

IMFMQI_MD_ACCOUNTINGTOKEN
9-29, 9-42, 9-53

IMFMQI_MD_APPLIDENTITYDATA
9-29, 9-42, 9-53

IMFMQI_MD_APPLORIGINDATA 9-30,
9-45

IMFMQI_MD_BACKOUTCOUNT 9-29,
9-42, 9-53

IMFMQI_MD_CODEDCHARSETID 9-28,
9-41, 9-52

IMFMQI_MD_CORRELID 9-29, 9-42,
9-53

IMFMQI_MD_ENCODING 9-28, 9-41,
9-52

IMFMQI_MD_EXPIRY 9-27, 9-51

IMFMQI_MD_FEEDBACK 9-27, 9-41,
9-52

IMFMQI_MD_FORMAT 9-28, 9-41, 9-52

IMFMQI_MD_GROUPID 9-30, 9-45

variables (continued)

IMFMQI_MD_MSGFLAGS 9-31, 9-45

IMFMQI_MD_MSGID 9-28, 9-42, 9-53

IMFMQI_MD_MSGSEQNUMBER 9-30,
9-45

IMFMQI_MD_MSGTYPE 9-27, 9-40, 9-51

IMFMQI_MD_OFFSET 9-30, 9-45

IMFMQI_MD_ORIGINALLENGTH 9-31,
9-45

IMFMQI_MD_PERSISTENCE 9-28, 9-42,
9-53

IMFMQI_MD_PRIORITY 9-28, 9-41, 9-53

IMFMQI_MD_PUTAPPLNAME 9-30,
9-44

IMFMQI_MD_PUTAPPLTYPE 9-30, 9-44

IMFMQI_MD_PUTDATE 9-30, 9-44

IMFMQI_MD_PUTTIME 9-30, 9-44

IMFMQI_MD_REPLYTOQ 9-29, 9-42,
9-53

IMFMQI_MD_REPLYTOQMGR 9-29,
9-42, 9-53

IMFMQI_MD_REPORT 9-27, 9-40, 9-51

IMFMQI_MD_STRUCID 9-26, 9-40, 9-50

IMFMQI_MD_USERIDENTIFIER 9-29,
9-42, 9-53

IMFMQI_MD_VERSION 9-26, 9-40, 9-50

IMFMQI_OD_ALTERNATESECURITYID
9-35, 9-50

IMFMQI_OD_ALTERNATEUSERID 9-35,
9-50

IMFMQI_OD_DYNAMICQNAME 9-35,
9-50

IMFMQI_OD_INVALIDDESTCOUNT
9-35, 9-50

IMFMQI_OD_KNOWNDESTCOUNT
9-35, 9-50

IMFMQI_OD_OBJECTNAME 9-34, 9-49

IMFMQI_OD_OBJECTQMGRNAME
9-35, 9-49

IMFMQI_OD_OBJECTRECOFFSET 9-35,
9-50

IMFMQI_OD_OBJECTRECPtr 9-35,
9-50

IMFMQI_OD_OBJECTTYPE 9-34, 9-49

variables (*continued*)

IMFMQI_OD_RECSPRESENT 9-35, 9-50

IMFMQI_OD_RESOLVEDQMGRNAME
9-35, 9-50

IMFMQI_OD_RESOLVEDQNAME 9-35,
9-50

IMFMQI_OD_RESPONSERECOFFSET
9-35, 9-50

IMFMQI_OD_RESPONSERECPTR 9-35,
9-50

IMFMQI_OD_STRUCID 9-34, 9-49

IMFMQI_OD_UNKNOWNDESTCOUNT
9-35, 9-50

IMFMQI_OD_VERSION 9-34, 9-49

IMS Information Header C-10, D-15

Message Descriptor C-1, D-5

Message Descriptor Extension C-17, D-22

MQSeries events 5-55

MQSeries messages,
AutoOPERATOR-supplied 5-55

POPTS 9-38

Reference Message Header C-20, D-25

SHARED 5-55

Transmission Queue Header C-12, D-17

Trigger Message C-5, D-10

Trigger Message 2 (character format) C-22,
D-27

TSO 9-60

Work Information Header C-19, D-24

viewing automation statistics 4-1

W

WAIT parameter, CMD 9-59

WAIT, Power Line parameter 7-2

Work Information Header Variables C-19

workstation panel

fields 4-5

primary commands 4-3

queue management 4-7

using 4-1

X

XMIT_Q_TYPE_ERROR 5-53

XMIT_Q_USAGE_ERROR 5-54

END USER LICENSE AGREEMENT NOTICE

BY OPENING THE PACKAGE, INSTALLING, PRESSING "AGREE" OR "YES" OR USING THE PRODUCT, THE ENTITY OR INDIVIDUAL ENTERING INTO THIS AGREEMENT AGREES TO BE BOUND BY THE FOLLOWING TERMS. IF YOU DO NOT AGREE WITH ANY OF THESE TERMS, DO NOT INSTALL OR USE THE PRODUCT, PROMPTLY RETURN THE PRODUCT TO BMC OR YOUR BMC RESELLER, AND IF YOU ACQUIRED THE LICENSE WITHIN 30 DAYS OF THE DATE OF YOUR ORDER CONTACT BMC OR YOUR BMC RESELLER FOR A REFUND OF LICENSE FEES PAID. IF YOU REJECT THIS AGREEMENT, YOU WILL NOT ACQUIRE ANY LICENSE TO USE THE PRODUCT.

This Agreement ("**Agreement**") is between the entity or individual entering into this Agreement ("You") and BMC Software Distribution, Inc., a Delaware corporation located at 2101 CityWest Blvd., Houston, Texas, 77042, USA or its affiliated local licensing entity ("BMC"). "You" includes you and your Affiliates. "Affiliate" is defined as an entity which controls, is controlled by or shares common control with a party. IF MORE THAN ONE LICENSE AGREEMENT COULD APPLY TO THE PRODUCT, THE FOLLOWING ORDER OF LICENSE AGREEMENT PRECEDENCE APPLIES: (1) WEB BASED LICENSE AGREEMENT WITH BMC, (2) WRITTEN LICENSE AGREEMENT WITH BMC, (3) SHRINK-WRAP LICENSE AGREEMENT WITH BMC PROVIDED WITH THE PRODUCT, AND (4) THIS ELECTRONIC LICENSE AGREEMENT WITH BMC. In addition to the restrictions imposed under this Agreement, any other usage restrictions contained in the Product installation instructions or release notes shall apply to Your use of the Product.

PRODUCT AND CAPACITY. "**Software**" means the object code version of the computer programs provided, via delivery or electronic transmission, to You. Software includes computer files, enhancements, maintenance modifications, upgrades, updates, bug fixes, and error corrections.

"**Documentation**" means all written or graphical material provided by BMC in any medium, including any technical specifications, relating to the functionality or operation of the Software.

"**Product**" means the Software and Documentation.

"**License Capacity**" means the licensed capacity for the Software with the pricing and other license defining terms, including capacity restrictions, such as tier limit, total allowed users, gigabyte limit, quantity of Software, and/or other capacity limitations regarding the Software. For licenses based on the power of a computer, You agree to use BMC's current computer classification scheme, which is available at <http://www.bmc.com> or can be provided to You upon request.

ACCEPTANCE. The Product is deemed accepted by You, on the date that You received the Product from BMC.

LICENSE. Subject to the terms of this Agreement, as well as Your payment of applicable fees, BMC grants You a non-exclusive, non-transferable, perpetual (unless a term license is provided on an order) license for each copy of the Software, up to the License Capacity, to do the following:

- (a) install the Software on Your owned or leased hardware located at a facility owned or controlled by You in the country where You acquired the license;
- (b) operate the Software solely for processing Your own data in Your business operations; and
- (c) make one copy of the Software for backup and archival purposes only (collectively a "**License**").

If the Software is designed by BMC to permit you to modify such Software, then you agree to only use such modifications or new software programs for Your internal purposes or otherwise consistent with the License. BMC grants You a license to use the Documentation solely for Your internal use in Your operations.

LICENSE UPGRADES. You may expand the scope of the License Capacity only pursuant to a separate agreement with BMC for such expanded usage and Your payment of applicable fees. There is no additional warranty period or free support period for license upgrades.

RESTRICTIONS: You agree to **NOT**:

- (a) disassemble, reverse engineer, decompile or otherwise attempt to derive any Software from executable code;
- (b) distribute or provide the Software to any third party (including without limitation, use in a service bureau, outsourcing environment, or processing the data of third parties, or for rental, lease, or sublicense); or
- (c) provide a third party with the results of any functional evaluation or benchmarking or performance tests, without BMC's prior written approval, unless prohibited by local law.

TRIAL LICENSE. If, as part of the ordering process, the Product is provided on a trial basis, then these terms apply: (i) this license consists solely of a non-exclusive, non-transferable evaluation license to operate the Software for the period of time specified from BMC or, if not specified, a 30 day time period ("**Trial Period**") only for evaluating whether You desire to acquire a capacity-based license to the Product for a fee; and (ii) Your use of the Product is on an AS IS basis without any warranty, and **BMC, ITS AFFILIATES AND RESELLERS, AND LICENSORS DISCLAIM ANY AND ALL WARRANTIES (INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT) AND HAVE NO LIABILITY WHATSOEVER RESULTING FROM THE USE OF THIS PRODUCT UNDER THIS TRIAL LICENSE ("Trial License").** BMC may terminate for its convenience a Trial License upon notice to You. When the Trial Period ends, Your right to use this Product automatically expires. If You want to continue Your use of the Product beyond the Trial Period, contact BMC to acquire a capacity-based license to the Product for a fee.

TERMINATION. This Agreement shall immediately terminate if You breach any of its terms. Upon termination, for any reason, You must uninstall the Software, and either certify the destruction of the Product or return it to BMC.

OWNERSHIP OF THE PRODUCT. BMC or its Affiliates or licensors retain all right, title and interest to and in the BMC Product and all intellectual property, informational, industrial property and proprietary rights therein. BMC neither grants nor otherwise transfers any rights of ownership in the BMC Product to You. Products are protected by applicable copyright, trade secret, and industrial and intellectual property laws. BMC reserves any rights not expressly granted to You herein.

CONFIDENTIAL AND PROPRIETARY INFORMATION. The Products are and contain valuable confidential information of BMC (“**Confidential Information**”). Confidential Information means non-public technical and non-technical information relating to the Products and Support, including, without limitation, trade secret and proprietary information, and the structure and organization of the Software. You may not disclose the Confidential Information to third parties. You agree to use all reasonable efforts to prevent the unauthorized use, copying, publication or dissemination of the Product.

WARRANTY. Except for a Trial License, BMC warrants that the Software will perform in substantial accordance with the Documentation for a period of one year from the date of the order. This warranty shall not apply to any problems caused by software or hardware not supplied by BMC or to any misuse of the Software.

EXCLUSIVE REMEDY. BMC’s entire liability, and Your exclusive remedy, for any defect in the Software during the warranty period or breach of the warranty above shall be limited to the following: BMC shall use reasonable efforts to remedy defects covered by the warranty or replace the defective Software within a reasonable period of time, or if BMC cannot remedy or replace such defective copy of the Software, then BMC shall refund the amount paid by You for the License for that Software. BMC’s obligations in this section are conditioned upon Your providing BMC prompt access to the affected Software and full cooperation in resolving the claim.

DISCLAIMER. EXCEPT FOR THE EXPRESS WARRANTIES ABOVE, THE PRODUCT IS PROVIDED “AS IS.” BMC, ITS AFFILIATES AND LICENSORS SPECIFICALLY DISCLAIM ALL OTHER WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. BMC DOES NOT WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT ALL DEFECTS CAN BE CORRECTED.

DISCLAIMER OF DAMAGES. IN NO EVENT IS BMC, ITS AFFILIATES OR LICENSORS LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES RELATING TO OR ARISING OUT OF THIS AGREEMENT, SUPPORT, AND/OR THE PRODUCT (INCLUDING, WITHOUT LIMITATION, LOST PROFITS, LOST COMPUTER USAGE TIME, AND DAMAGE OR LOSS OF USE OF DATA), EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND IRRESPECTIVE OF ANY NEGLIGENCE OF BMC OR WHETHER SUCH DAMAGES RESULT FROM A CLAIM ARISING UNDER TORT OR CONTRACT LAW.

LIMITS ON LIABILITY. BMC’S AGGREGATE LIABILITY FOR DAMAGES IS LIMITED TO THE AMOUNT PAID BY YOU FOR THE LICENSE TO THE PRODUCT.

SUPPORT. If Your order includes support for the Software, then BMC agrees to provide support (24 hours a day/7 days a week) (“**Support**”). You will be automatically re-enrolled in Support on an annual basis unless BMC receives notice of termination from You as provided below. There is a free support period during the one year warranty period.

(a) **Support Terms.** BMC agrees to make commercially reasonable efforts to provide the following Support: (i) For malfunctions of supported versions of the Software, BMC provides bug fixes, patches or workarounds in order to cause that copy of the Software to operate in substantial conformity with its then-current operating specifications; and (ii) BMC provides new releases or versions, so long as such new releases or versions are furnished by BMC to all other enrolled Support customers without additional charge. BMC may refuse to provide Support for any versions or releases of the Software other than the most recent version or release of such Software made available by BMC. Either party may terminate Your enrollment in Support upon providing notice to the other at least 30 days prior to the next applicable Support anniversary date. If You re-enroll in Support, BMC may charge You a reinstatement fee of 1.5 times what You would have paid if You were enrolled in Support during that time period.

(b) **Fees.** The annual fee for Support is 20% of the Software’s list price less the applicable discount or a flat capacity based annual fee. BMC may change its prices for the Software and/or Support upon at least 30 days notice prior to Your support anniversary date.

VERIFICATION. If requested by BMC, You agree to deliver to BMC periodic written reports, whether generated manually or electronically, detailing Your use of the Software in accordance with this Agreement, including, without limitation, the License Capacity. BMC may, at its expense, perform an audit, at your facilities, of Your use of the Software to confirm Your compliance with the Agreement. If an audit reveals that You have underpaid fees, You agree to pay such underpaid fees. If the underpaid fees exceed 5% of the fees paid, then You agree to also pay BMC’s reasonable costs of conducting the audit.

EXPORT CONTROLS. You agree not to import, export, re-export, or transfer, directly or indirectly, any part of the Product or any underlying information or technology except in full compliance with all United States, foreign and other applicable laws and regulations.

GOVERNING LAW. This Agreement is governed by the substantive laws in force, without regard to conflict of laws principles: (a) in the State of New York, if you acquired the License in the United States, Puerto Rico, or any country in Central or South America; (b) in the Province of Ontario, if you acquired the License in Canada (subsections (a) and (b) collectively referred to as the “**Americas Region**”); (c) in Singapore, if you acquired the License in Japan, South Korea, Peoples Republic of China, Special Administrative Region of Hong Kong, Republic of China, Philippines, Indonesia, Malaysia, Singapore, India, Australia, New Zealand, or Thailand (collectively, “**Asia Pacific Region**”); or (d) in the Netherlands, if you acquired the License in any other country not described above. The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed in its entirety.

ARBITRATION. ANY DISPUTE BETWEEN YOU AND BMC ARISING OUT OF THIS AGREEMENT OR THE BREACH OR ALLEGED BREACH, SHALL BE DETERMINED BY BINDING ARBITRATION CONDUCTED IN ENGLISH. IF THE DISPUTE IS INITIATED IN THE AMERICAS REGION, THE ARBITRATION SHALL BE HELD IN NEW YORK, U.S.A., UNDER THE CURRENT COMMERCIAL OR INTERNATIONAL, AS APPLICABLE, RULES OF THE AMERICAN ARBITRATION ASSOCIATION. IF THE DISPUTE IS INITIATED IN A COUNTRY IN THE ASIA PACIFIC REGION, THE ARBITRATION SHALL BE HELD IN SINGAPORE, SINGAPORE UNDER THE CURRENT UNCITRAL ARBITRATION RULES. IF THE DISPUTE IS INITIATED IN A COUNTRY OUTSIDE OF THE AMERICAS REGION OR ASIA PACIFIC REGION, THE ARBITRATION SHALL BE HELD IN AMSTERDAM, NETHERLANDS UNDER THE CURRENT UNCITRAL ARBITRATION RULES. THE COSTS OF THE ARBITRATION SHALL BE BORNE EQUALLY PENDING THE ARBITRATOR’S AWARD. THE AWARD RENDERED SHALL BE FINAL AND BINDING UPON THE PARTIES AND SHALL NOT BE SUBJECT TO APPEAL TO ANY COURT, AND MAY BE ENFORCED IN ANY COURT OF COMPETENT JURISDICTION. NOTHING IN THIS AGREEMENT SHALL BE DEEMED AS PREVENTING EITHER PARTY FROM SEEKING INJUNCTIVE RELIEF FROM ANY COURT HAVING JURISDICTION OVER THE PARTIES

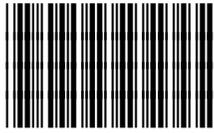
AND THE SUBJECT MATTER OF THE DISPUTE AS NECESSARY TO PROTECT EITHER PARTY'S CONFIDENTIAL INFORMATION, OWNERSHIP, OR ANY OTHER PROPRIETARY RIGHTS. ALL ARBITRATION PROCEEDINGS SHALL BE CONDUCTED IN CONFIDENCE, AND THE PARTY PREVAILING IN ARBITRATION SHALL BE ENTITLED TO RECOVER ITS REASONABLE ATTORNEYS' FEES AND NECESSARY COSTS INCURRED RELATED THERETO FROM THE OTHER PARTY.

U.S. GOVERNMENT RESTRICTED RIGHTS. The Software under this Agreement is "commercial computer software" as that term is described in 48 C.F.R. 252.227-7014(a)(1). If acquired by or on behalf of a civilian agency, the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of this Agreement as specified in 48 C.F.R. 12.212 (Computer Software) and 12.211 (Technical Data) of the Federal Acquisition Regulations ("**FAR**") and its successors. If acquired by or on behalf of any agency within the Department of Defense ("**DOD**"), the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of this Agreement as specified in 48 C.F.R. 227.7202 of the DOD FAR Supplement and its successors.

MISCELLANEOUS TERMS. You agree to pay BMC all amounts owed no later than 30 days from the date of the applicable invoice, unless otherwise provided on the order for the License to the Products. You will pay, or reimburse BMC, for taxes of any kind, including sales, use, duty, tariffs, customs, withholding, property, value-added (VAT), and other similar federal, state or local taxes (other than taxes based on BMC's net income) imposed in connection with the Product and/or the Support. This Agreement constitutes the entire agreement between You and BMC and supersedes any prior or contemporaneous negotiations or agreements, whether oral, written or displayed electronically, concerning the Product and related subject matter. No modification or waiver of any provision hereof will be effective unless made in a writing signed by both BMC and You. You may not assign or transfer this Agreement or a License to a third party without BMC's prior written consent. Should any provision of this Agreement be invalid or unenforceable, the remainder of the provisions will remain in effect. The parties have agreed that this Agreement and the documents related thereto be drawn up in the English language. Les parties exigent que la présente convention ainsi que les documents qui s'y rattachent soient rédigés en anglais.

SW Click EULA 071102

Notes



42111