



BEA MessageQ

Configuration Guide for OpenVMS

BEA MessageQ for OpenVMS Version 5.0
Document Edition 5.0
March 2000

Copyright

Copyright © 2000 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, ObjectBroker, TOP END, and TUXEDO are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Connect, BEA Manager, BEA MessageQ, Jolt, M3, and WebLogic are trademarks of BEA Systems, Inc.

All other company names may be trademarks of the respective companies with which they are associated.

BEA MessageQ Configuration Guide for OpenVMS

Document Edition	Date	Software Version
5.0	March 2000	BEA MessageQ Version 5.0

Contents

Preface

1. Preparing for BEA MessageQ Implementation

BEA MessageQ Basic Concepts and Terms	1-1
Supported Network Protocols	1-2
Message Queues	1-2
BEA MessageQ Server Processes	1-3
Naming	1-6
Global Memory	1-6
BEA MessageQ / BEA TUXEDO Bridge.....	1-7
Configuring Distributed Systems Using BEA MessageQ.....	1-7
Design Paradigms.....	1-8
Traditional Functional Model	1-8
Object-Oriented Methodology	1-9
Determining Queue Sizes.....	1-9
Simulating Worst-Case Load Scenario	1-11
Failover Provisions.....	1-11
Design Summary	1-12

2. Defining the Message Queuing Environment

Preparing to Define the BEA MessageQ Environment.....	2-2
BEA MessageQ Manager Account Privileges	2-2
Basic Configuration Steps.....	2-2
Creating a Message Queuing Group.....	2-4
The BEA MessageQ Message Queuing Network.....	2-4
Assigning Bus and Group IDs.....	2-5
Creating Message Queuing Groups.....	2-6

Default Directory Names.....	2-7
Connecting to the BEA MessageQ Logical Name Table.....	2-8
Editing DMQ\$INIT to Configure a Group.....	2-10
Setting Servers and Configuring Groups in the Profile Section.....	2-12
Profile Section Parameters	2-13
Default Namespace Path Definition	2-15
Setting Network Connections in the Cross-group Section.....	2-15
Configuring Client Library Server in the CLS section	2-16
Configuring Local Buffer Pools in the Buffer section	2-16
Defining Queues in the Queue Configuration Section.....	2-17
Setting Broadcast System Parameters in the SBS Section.....	2-18
Setting Message Recovery System Parameters in the MRS Section	2-19
Assigning the Groups in which the Naming Agent Will Run.....	2-20
Starting and Stopping BEA MessageQ Groups.....	2-20
Using DMQ\$STARTUP to Invoke BEA MessageQ	2-21
Invoking DMQ\$STARTUP without Starting the Servers.....	2-22
Starting Groups Under an Older Version of BEA MessageQ.....	2-23
Starting Application Programs in a Message Queuing Group	2-23
Application Startup Process	2-23
Two Methods for Connecting.....	2-24
Detaching a Process with DCL Context.....	2-24
Detaching a Process without DCL Context.....	2-25
Shutting Down a Running BEA MessageQ Group	2-26
Additional Configuration Tasks	2-27
Configuring Event Logging.....	2-28
Adding Queue Names to Network-wide Namespace.....	2-28
Changing Parameters in the Running Group.....	2-29
Deleting Groups.....	2-31
Defining Message Type and Class Codes	2-32
BEA MessageQ Hints and Tips.....	2-32
Modifying Your Default Editor.....	2-32
Defining BEA MessageQ Symbols.....	2-32
Startup Synchronization	2-33
Redirecting Configuration and Log Files.....	2-33

3. Configuring Cross-group Connections

Connecting to Other Message Queuing Groups	3-2
Configuring the Cross-group Connection Table	3-2
Cross-group Connection Table Overview	3-3
Loading the Configuration Data	3-3
Using Cross-group Connection Table Fields	3-3
Table Entry Guidelines	3-6
Using Message Routing	3-8
Routing Tables	3-8
Configuring and Loading the Tables	3-9
DMQ\$INIT Routing Section	3-10
Selecting the TCP/IP Link Driver	3-11
Sharing Group Configuration Files	3-12
Suppressing DECnet Intrusion Alarms	3-13
Configuring DMQ\$GMT_OFFSET for Network Communications	3-15

4. Configuring Message Queues and Global Memory

Configuring Message Queues	4-1
Queue Configuration Table Parameters	4-3
Queue Configuration Table Rules	4-5
Configuring Global Memory	4-7
How BEA MessageQ Uses Global Memory	4-7
Using the Buffer Pool Configuration Table	4-8

5. Configuring Message Recovery

How Message Recovery Services Work	5-2
Using Recoverable Journal Files	5-2
Using Auxiliary Journal Files	5-3
Using the Dead Letter Journal	5-4
Using the Post Confirmation Journal	5-5
Retrieving Journalled Messages	5-5
Starting MRS and JRN Servers	5-6
Configuring MRS and JRN Servers	5-6
Setting Parameters for MRS and JRN	5-7
How BEA MessageQ Manages Destination Queue Files (DQFs)	5-10

Specifying the Location of MRS Files	5-11
Sizing MRS File Space	5-11
Sizing the Amount of Recovery File Space	5-11
Sizing the Journal File Chunk Size	5-13
Sizing MRS Server In-Memory Data Structures at Startup	5-13
Reducing MRS Server Startup Time in Large-Scale Applications	5-14
MRS Internal Operation Tracing at Startup	5-14
Confirming Message Removal from the Recovery System	5-15
Using the CONFIRM_STYLE Attribute.....	5-15
Controlling Recovery System Response with DMQ\$BLOCKING_CONFIRM.....	5-16
Using RCVR_ONLY_CONFIRM Parameter	5-17

6. Setting Up Selective Broadcasting

Overview of Selective Broadcasting	6-1
Starting the SBS Server	6-2
Configuring Broadcasting.....	6-3
Specifying Multipoint Outbound Target (MOT) Addresses	6-3
Configuring Optimized Ethernet Mode.....	6-4
Broadcasting Requirements and Restrictions	6-9
SBS Interoperability	6-10
Ethernet Broadcast Recovery Methods	6-11
Configuring Optimized Ethernet Mode.....	6-11
Dual-rail Mode	6-12
RP/ETH Retransmission Protocol	6-12
RP/ETH Receive Silo Rules.....	6-13
RP/ETH Examples	6-14
RP/ETH Configuration.....	6-16

7. Creating Global Names

Configuring BEA MessageQ Global Naming	7-1
Configure Groups to Run or Use the Naming Agent	7-2
Configure a Lightweight Namespace	7-2
Default Namespace Path Definition	7-4
Configure a Default Namespace Path for Each Group.....	7-4

Define the Queue Names in the Group Initialization File.....	7-5
Using DNS with Global Naming	7-7
Managing the Global Namespace	7-8
Viewing a Group's Cache	7-8
Global Name Examples	7-9
Defining Type and Class Codes	7-12

8. Configuring the BEA MessageQ Client Library Server

BEA MessageQ Client Library Server Overview	8-2
Client Library Server Installation and Transport Support	8-3
Configuring the CLS section of DMQ\$INIT.TXT File	8-4
CLS Endpoints	8-5
Setting Maximum Number of Clients for CLS	8-6
Configuring One Client for Each CLS Server Option	8-7
Restricting Remote Access to CLS	8-8
Special Queue Configuration Issues.....	8-9
Errors Attaching to Undefined Queues	8-11
Starting and Stopping CLS Manually.....	8-11
Starting CLS from the Manager Utility.....	8-12
Starting CLS from DCL	8-13
Stopping CLS from the Manager Utility	8-13
CLS Event Logging and Tracing.....	8-15

9. BEA MessageQ Main Menu and Utilities

Using the BEA MessageQ Main Menu.....	9-2
Running the IVP	9-3
Starting and Shutting Down the COM Server	9-3
Verifying Buffer Pool Configuration (DMQ\$LLS_VERIFY).....	9-4
Testing BEA MessageQ Communications with DMQ\$LOOP	9-4
Running DMQ\$LOOP	9-5
Testing BEA MessageQ Services with DMQ\$TEST.....	9-6
Running the CUSTOMIZE Command Procedure.....	9-6
Customization Options	9-6
About the DMQ\$BOOT Command Procedure.....	9-8
About the DMQ\$INIT.TXT File.....	9-9

About the DMQ\$TYPCLS.TXT File	9-9
Running the DMQ\$LOADER Utility.....	9-9
Starting the LOADER Utility	9-9
Restrictions on DMQ\$LOADER	9-9
Changing Bus and Group Numbers.....	9-10
Shutting Down BEA MessageQ.....	9-10

10. Using BEA MessageQ System Management Utilities

Major System Management Tasks	10-1
Using the Monitor Utility (DMQ\$MONITOR).....	10-4
Displaying Queue Counters.....	10-5
Displaying Queue Quotas.....	10-5
Displaying COM Server Status	10-6
Displaying Queue-Specific Status	10-7
Resetting COM Server Counters	10-7
Displaying the Routing Table.....	10-7
Shutting Down COM Server Process	10-7
Displaying Link Summary Information	10-8
Displaying Link Detail Information	10-8
Displaying Link Connect Table.....	10-9
Displaying Group Detail Information	10-9
Resetting Cross-Group Connections	10-9
Displaying Remote Groups	10-9
Using the System Management Utility (DMQ\$MGR_UTILITY).....	10-10
Displaying Queue Summary Information	10-10
Displaying Queue Detail Information	10-10
Flushing Queues	10-10
Stopping Queues.....	10-11
Starting Queues	10-11
Forcing a Process to Exit.....	10-11
Redirecting Status and Trace Output.....	10-12
Enabling Tracing Prior to Starting a Program.....	10-12
Enabling Tracing When a Program is Running	10-13
Event Logging	10-14
Directing Error and Status Messages Prior to Starting a Program ..	10-14

Redirecting Error and Status Logging When a Program is Running	10-14
Using Event Log Control	10-14
Storing Event Data	10-15
Switching Event Log Files	10-15
Displaying Group Name Table	10-15
Journal Controls	10-16
Managing DQF and SAF Journals	10-16
Managing DLJ and PCJ Journals	10-17
Managing the Naming Agent and the Namespace	10-18
Displaying Information About Names and the Namespace	10-19
Managing Address and Name Associations	10-21
Removing Names from the Namespace	10-22

11. Sizing and Tuning the BEA MessageQ Environment

Sizing and Tuning Processes	11-2
Virtual Memory	11-2
Global Memory	11-3
I/O Channels	11-3
Files	11-4
Network Resources	11-4
Other System Resources and Quotas	11-4
Modifying DMQ\$SET_SERVER_QUOTAS.COM	11-5
Allocating Virtual Memory for BEA MessageQ Servers	11-6
Modeling Virtual Memory Needs	11-6
Performing Testing	11-7
Modeling Memory Usage for Each BEA MessageQ Server	11-7
Example Memory Allocation Model for the MRS Server	11-10
Global Memory	11-12
Tuning TCP/IP for BEA MessageQ	11-13
Approximating the Nonpaged Pool Needs	11-13
Computing the Number of TCP/IP Sockets	11-13
Configuring BEA MessageQ for Large Messages	11-14
Maximum Message Size	11-14
Message Pool Configuration	11-14

Buffer Pool Parameter	11-16
Queue Quota	11-16
Global Section Size	11-16
Timeouts	11-17
Message Recovery Services	11-17

12. Managing a BEA MessageQ Environment

BEA MessageQ Error Logging	12-2
BEA MessageQ Output	12-2
BEA MessageQ Stream Output.....	12-2
Stream Destinations.....	12-3
Stream Switches	12-4
The Event Logger.....	12-4
Console Output.....	12-5
In-Memory Logging	12-5
Enabling Tracing	12-5
BEA MessageQ Servers	12-6
BEA MessageQ Server Output Files.....	12-6
BEA MessageQ Server Logging/Debugging	12-7
Tools for Troubleshooting BEA MessageQ Problems	12-8
Server Log Messages.....	12-9
API Return Status Values	12-10
Utility Programs	12-10
DMQ\$TEST	12-10
DMQ\$LOOP	12-11
DMQ\$LLS_VERIFY	12-11
DMQ\$MONITOR	12-11
DMQ\$MGR_UTILITY	12-11
DMQ\$SCAN_SYSTEM_FOR_DMQ.COM.....	12-11
Troubleshooting Procedures	12-12
Basic Troubleshooting Tasks	12-12
Diagnosing Application Errors.....	12-12
Enabling Server Tracing.....	12-13
Verifying BEA MessageQ Group Startup.....	12-13
Troubleshooting BEA MessageQ Startup Problems	12-15

Group Startup Failure.....	12-15
Additional Group Startup Problems.....	12-17
Troubleshooting Problems with Running Groups and Queues.....	12-18
BEA MessageQ Connectivity Troubleshooting.....	12-18
BEA MessageQ Recovery Troubleshooting.....	12-20
BEA MessageQ Application Troubleshooting.....	12-22

13. BEA MessageQ Security

Rights Identifiers Used By BEA MessageQ.....	13-1
Defining Access Control on Queues.....	13-2
Securing Readout of Permanent Queues.....	13-3
Securing the Creation of Temporary Queues.....	13-3
Setting the Global Section Protection Mask.....	13-4
Defining Protection Mask Logical Name.....	13-4
Setting ACLs for Global Sections.....	13-5
Controlling Network Access to Queuing Group.....	13-6

14. Managing Failover

Failover Mechanisms.....	14-1
Group Failover.....	14-1
Recoverable Queue Failover.....	14-2
Managing and Planning for Failover.....	14-3
Cold Failover.....	14-3
Hot Failover to a Running Shadow Group.....	14-5
Automatic Synchronized Cluster Failover.....	14-6
Implementing Automatic Synchronized Failover.....	14-6
Non-automatic Hot Failover.....	14-7
Implementing Non-Automatic Hot Failover.....	14-8
Detecting a Failure While the Failing Group Is Still Running.....	14-10
Programming Considerations.....	14-10
Failover of Recoverable Messaging to and from a Single Target.....	14-11
Redirecting Recoverable Data Streams.....	14-11
Merging Recoverable Data Streams.....	14-13
Supporting Failover of SAF Files.....	14-15

15. Configuring the BEA MessageQ Bridge

Enabling the Messaging Bridge.....	15-3
Target, Queue Space, and Queue Name Mapping	15-5
TUXEDO Queue Space to BEA MessageQ Group Name.....	15-6
TUXEDO Queue to BEA MessageQ Queue.....	15-7

A. Sample DMQ\$INIT.TXT File

B. Sample DMQ\$TYPCLS.TXT File

C. Parameter Tuning Guidelines

D. Directories and Logical Names

Directories	D-1
Logical Names	D-2

E. Error Log Messages

COM Server and Link Driver Error Log Messages.....	E-1
Additional Information on Link Drivers Error Status Reporting	E-36
MRS Server Error Log Messages	E-37
SBS Error Log Messages	E-56

Index

Preface

Purpose of This Document

This document gives instructions for configuring the BEA MessageQ for OpenVMS software.

Who Should Read This Document

This document is intended for BEA MessageQ for OpenVMS software installers and system administrators.

How This Document Is Organized

The *BEA MessageQ Configuration Guide for OpenVMS* provides instructions on how to configure your BEA MessageQ environment on OpenVMS systems.

The *BEA MessageQ Configuration Guide for OpenVMS* is organized as follows:

- Chapter 1, “Preparing for BEA MessageQ Implementation,” discusses basic terms and concepts as well as the planning necessary for a successful implementation of BEA MessageQ software on OpenVMS systems.
- Chapter 2, “Defining the Message Queuing Environment,” describes how to configure MessageQ during initial system configuration.

-
- Chapter 3, “Configuring Cross-group Connections,” describes how to establish communication among applications running in different message queuing groups in a networked environment.
 - Chapter 4, “Configuring Message Queues and Global Memory,” describes how to configure the message queues and global sections that MessageQ uses to send, store, and retrieve messages.
 - Chapter 5, “Configuring Message Recovery,” describes how to configure MessageQ systems to use recoverable messaging.
 - Chapter 6, “Setting Up Selective Broadcasting,” describes how to configure MessageQ systems to use message broadcasting.
 - Chapter 7, “Creating Global Names,” describes how to use MessageQ shared local and global symbols.
 - Chapter 8, “Configuring the BEA MessageQ Client Library Server,” describes how to communicate with MessageQ PC-based clients.
 - Chapter 9, “BEA MessageQ Main Menu and Utilities,” describes the MessageQ Main Menu, its major options and features.
 - Chapter 10, “Using BEA MessageQ System Management Utilities,” describes the MessageQ system management and monitoring utilities.
 - Chapter 11, “Sizing and Tuning the BEA MessageQ Environment,” describes how to tune MessageQ software to support your application.
 - Chapter 12, “Managing a BEA MessageQ Environment,” describes steps for troubleshooting MessageQ problems.
 - Chapter 13, “BEA MessageQ Security,” describes MessageQ security features.
 - Chapter 14, “Managing Failover,” describes how to perform failover for a message queuing group.
 - Chapter 15, “Configuring the BEA MessageQ Bridge,” describes how to configure the bridge enabling message exchange between BEA MessageQ and BEA TUXEDO.
 - Appendix A, “Sample DMQ\$INIT.TXT File,” provides a sample MessageQ initialization file (DMQ\$INIT.TXT).

-
- Appendix B, “Sample DMQ\$TYPCLS.TXT File,” provides a sample MessageQ message definition file (DMQ\$TYPCLS.TXT).
 - Appendix C, “Parameter Tuning Guidelines,” provides guidelines for tuning OpenVMS and MessageQ parameters.
 - Appendix D, “Directories and Logical Names,” lists MessageQ directories and logical names.
 - Appendix E, “Error Log Messages,” lists warning and error messages logged by the COM server, SBS server, MRS server, and link drivers.

How to Use This Document

This document, *BEA MessageQ Configuration Guide for OpenVMS*, is designed primarily as an online, hypertext document. If you are reading this as a paper publication, note that to get full use from this document you should install and access it as an online document via a Web browser.

The following sections explain how to view this document online, and how to print a copy of this document.

Opening the Document in a Web Browser

To access the online version of this document, open the following HTML file in a Web browser:

```
/doc/bmq/v5_0/base/vmsinst/index.htm
```

Note: The online documentation requires a Web browser that supports HTML version 3.0. Netscape Navigator version 3.0 or Microsoft Internet Explorer version 3.0 or later is recommended.

Printing from a Web Browser

You can print a copy of this document, one file at a time, from the Web browser. Before you print, make sure that the chapter or appendix you want is displayed and *selected* in your browser. (To select a chapter or appendix, click anywhere inside the chapter or appendix you want to print. If your browser offers a Print Preview feature, you can use the feature to verify which chapter or appendix you are about to print.)

The BEA MessageQ Online Documentation CD also includes Adobe Acrobat PDF files of all of the online documents. You can use the Adobe Acrobat Reader to print all or a portion of each document.

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary in the <i>BEA MessageQ Introduction to Message Queuing</i> .
Ctrl+Tab	Indicates that you must press two or more keys sequentially.
<i>italics</i>	Indicate emphasis or book titles.
monospace text	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard. <i>Examples:</i> <pre>#include stdio pams_attach_q \bmq\lu62_40a\include .htm bmq.doc BITMAP float</pre>

Convention	Item
monospace boldface text	Identifies significant words in code. <i>Example:</i> <code>put_msg(msg_ptr, class, type)</code>
<i>monospace</i> <i>italic</i> <i>text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> LPT1 PATH OR
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[]	Indicates optional items in a syntax line. The brackets themselves should never be typed. <i>Example:</i> <code>int32 pams_get_msg (msg_area, priority ... [<i>sel_filter</i>] [psb] [<i>show_buffer</i>]...)</code>
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. <i>Example:</i> <code>int32 pams_get_msg (msg_area, priority ... [<i>sel_filter</i>] [psb] [<i>show_buffer</i>]...)</code>
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Related Documentation

The following sections list the documentation provided with the BEA MessageQ for OpenVMS software.

BEA MessageQ for OpenVMS Documentation

The BEA MessageQ for OpenVMS information set consists of the following documents:

BEA MessageQ Introduction to Message Queuing

BEA MessageQ Programmer's Guide

BEA MessageQ Installation Guide for OpenVMS

BEA MessageQ Configuration Guide for OpenVMS

BEA MessageQ System Messages

BEA MessageQ FML Programmer's Guide

BEA MessageQ Reference Manual

BEA MessageQ Client for OpenVMS User's Guide

Note: The BEA MessageQ Online Documentation CD also includes Adobe Acrobat PDF files of all of the online documents. You can use the Adobe Acrobat Reader to print all or a portion of each document.

Contact Information

The following sections provide information about how to obtain support for the documentation and software.

Documentation Support

If you have questions or comments on the documentation, you can contact the BEA Information Engineering Group by e-mail at **docsupport@beasys.com**. (For information about how to contact Customer Support, refer to the following section.)

Customer Support

If you have any questions about this version of BEA MessageQ, or if you have problems installing and running BEA MessageQ, contact BEA Customer Support through BEA WebSupport at www.beasys.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages



1 Preparing for BEA MessageQ Implementation

This chapter discusses BEA MessageQ basic terms and concepts as well as the planning necessary for a successful implementation of BEA MessageQ software on OpenVMS systems.

The following topics are covered in this chapter:

- ◆ BEA MessageQ Basic Concepts and Terms
- ◆ Configuring Distributed Systems Using BEA MessageQ

BEA MessageQ Basic Concepts and Terms

The basic unit of BEA MessageQ system management is the message queuing group. A message queuing group is a collection of message queues that are used by an application and that share access to a local interprocess communications (IPC) facility and BEA MessageQ Servers. A message queuing group can support more than one application. Message queuing groups are connected to one another using cross-group connections over network links.

Supported Network Protocols

BEA MessageQ has full heterogeneous communications capability for directing BEA MessageQ messages across message queuing groups to any supported platforms.

BEA MessageQ for OpenVMS software resides on top of DECnet and Transmission Control Protocol/Internet Protocol (TCP/IP) networking software and various intra-CPU communications mechanisms. On Compaq VAX and Alpha systems, BEA MessageQ for OpenVMS supports communications using DECnet, TCP/IP, LU6.2 protocol, and Ethernet.

Message Queues

The **BEA MessageQ message queuing bus** provides the interprocess communications vehicle that enables applications to exchange information using queued messaging. The message queuing bus is a set of BEA MessageQ message queuing groups that are configured to communicate with each other.

A message queue provides an area for an application to store and retrieve messages. A message queue is configured by the application developer and managed by BEA MessageQ. To receive BEA MessageQ messages, an application must be associated with at least one queue.

Message queues can be thought of as attachment points on the message queuing bus. A message queue is a physical resource with a unique ID within a group and can be permanent or temporary. A permanent queue exists whether or not a process is attached to it. A temporary queue exists only when a process is attached to the message queuing bus.

BEA MessageQ supports three types of message queues: primary, secondary, and multireader.

◆ **Primary Queue (PQ)**

The primary queue acts as the main mailbox for a user process. An application can have only one primary queue, although it may be associated with queues of other types. When an application reads a message from its primary queue, or any other queue, the message is removed from the queue. Messages are read in first-in/first-out (FIFO) order unless another order is specified.

◆ **Secondary Queue (SQ)**

The secondary queue acts as an alternate mailbox for the user process to receive messages.

◆ **Multireader Queue (MRQ)**

A multireader queue acts as a central mailbox that is used by several user processes (applications). Many user processes can simultaneously attach and read messages from a multireader queue.

BEA MessageQ Server Processes

The BEA MessageQ environment consists of two kinds of processes:

- ◆ **User processes** are the application programs that communicate using BEA MessageQ software. User processes running on the same system can share the same message queuing group or can reside in different groups. User processes running on different systems are defined in different groups. Remote applications communicate through **cross-group** communications.
- ◆ **BEA MessageQ Servers** on OpenVMS systems are detached processes that provide services in addition to basic message queuing. They implement portions of the BEA MessageQ application programming interface (API) or communicate with other BEA MessageQ message queuing groups.

All user processes in a BEA MessageQ for OpenVMS message queuing group share several sections of global memory, run-time libraries (RTLs), and are served by the following BEA MessageQ Server processes:

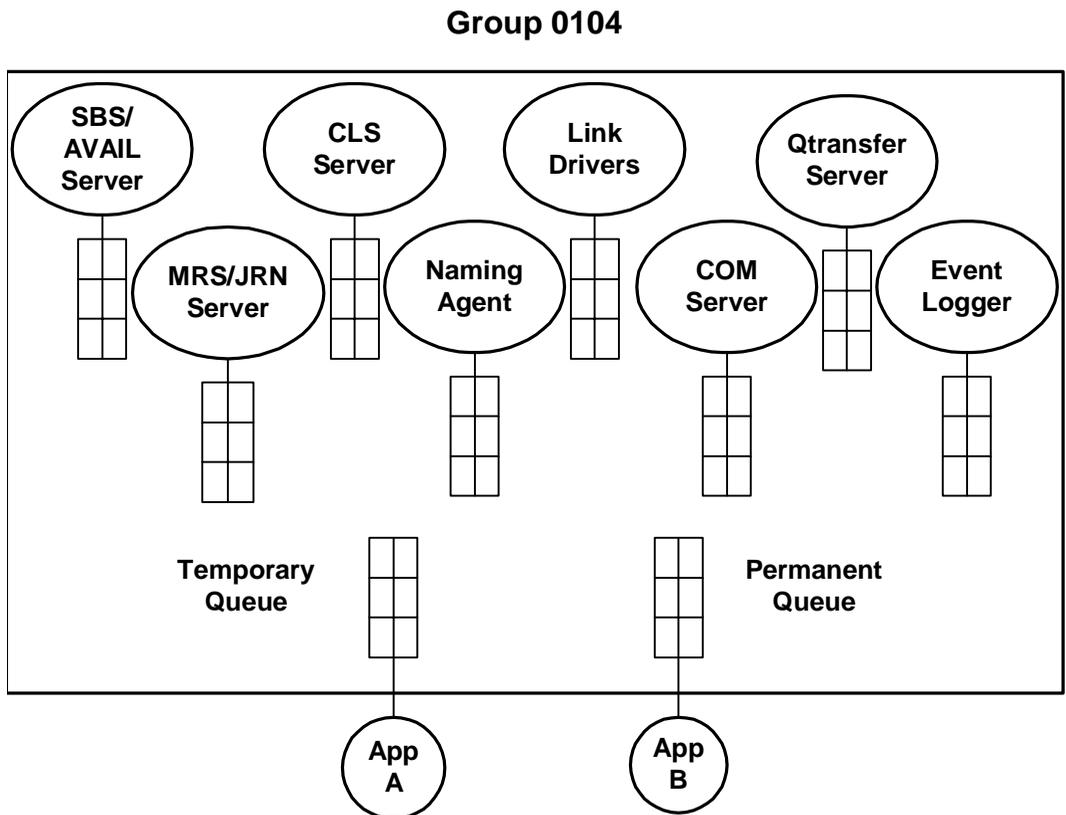
- ◆ COM Server—central resource control
- ◆ SBS Server—broadcast and AVAIL services
- ◆ MRS Server—recoverable messaging
- ◆ CLS Server—client connections
- ◆ Link driver—cross-group communications
- ◆ Event logger—status and error logging
- ◆ Journal Server—auxiliary journal processing

1 Preparing for BEA MessageQ Implementation

- ◆ Naming Agent—global naming and runtime binding of queue names to queue addresses
- ◆ Qtransfer Server—manages transfer of recoverable messages to a different queue

Figure 1-1 shows the servers and other process components of a message queuing group.

Figure 1-1 Components of a BEA MessageQ Message Queuing Group



- ◆ COM Server:
The COM Server creates and maintains the message queuing environment for the message queuing group and provides many of the BEA MessageQ

message-based services. The COM Server also coordinates the actions of the link drivers which provide cross-group communication. It contains a DECnet link driver that is used to communicate to other OpenVMS systems and older versions of BEA MessageQ software.

◆ **Message Recovery Services (MRS):**

The Message Recovery Services Server manages the disk storage required to handle recoverable message traffic. The BEA MessageQ message recovery system guarantees message delivery in the event of system, network, or application failures. Messages designated as recoverable are directed to an MRS Server for storage and removal from storage when delivery is confirmed by a user or another MRS Server.

◆ **Selective Broadcast Services (SBS):**

The Selective Broadcast Services Server controls the broadcasting of data between an application program and multiple receiving application programs. In addition, the SBS Server provides AVAIL/UNAVAIL message-based services.

◆ **Client Library Server (CLS):**

The Client Library Server controls the access of the Windows-based client applications to the message queuing functionality available using BEA MessageQ for UNIX, Windows NT, or OpenVMS.

◆ **Link Drivers:**

A link driver provides transparent communications over a particular network transport. BEA MessageQ for OpenVMS supports cross-group communication in both DECnet and TCP/IP network environments.

◆ **Journal Server:**

The Journal Server controls journal management functions handling opening, closing, and dumping of all auxiliary journal files.

◆ **Naming Agent:**

The Naming Agent maintains the namespace for name-to-queue address translation and performs the runtime lookup of the queue address when an application refers to a queue by name.

◆ **Event Logger:**

The event logger manages the common event log `EVL_LOG`. It serves as a single repository for logging all BEA MessageQ events. All entries are time stamped, and the file is sortable by various keywords, if error messages from a particular source needs to be isolated. The `EVL_LOG` can be closed, and moved and a new log started, from the `DMQ$MGR_UTILITY` menu.

◆ **Qtransfer Server**

The Queue Transfer Server manages the transfer of recoverable messages to a different queue.

Naming

Naming is a BEA MessageQ capability that enables applications to refer to queues by name instead of using their physical address in the BEA MessageQ environment. Using naming separates applications from the specifics of the network environment and enables system managers to make configuration changes without requiring developers to change the applications.

BEA MessageQ names can be defined to have a local or global scope. Local names are visible only to applications running in a particular group. Global names are available for use by any application attached to the message queuing group.

The BEA MessageQ process that supports the global naming capability is called the Naming Agent. The Naming Agent is responsible for creating and managing the name space of global name definitions and for name-to-queue address translation at runtime. In addition to its built-in ability to support global naming, the BEA MessageQ naming feature can also use a Distributed Name Server (DNS) to provide global naming capabilities.

Global Memory

The BEA MessageQ global memory provides storage for queue addresses and allocates memory space for message queuing groups and message buffers.

BEA MessageQ / BEA TUXEDO Bridge

BEA MessageQ V5.0 include a messaging bridge that allows the exchange of messages between BEA MessageQ V5.0 and BEA TUXEDO V6.5. This exchange of messages is made possible by two TUXEDO servers that are included in the BEA MessageQ kit and that run on the same machine as BEA MessageQ: TMQUEUE_BMQ and TMQFORWARD_BMQ. The messaging bridge is available only on OpenVMS Alpha 7.1 systems.

Configuring Distributed Systems Using BEA MessageQ

The basic unit of BEA MessageQ system management is the message queuing group. Message queuing groups consist of several message queues and share access to a common set of BEA MessageQ Servers. Message queuing groups are connected to one another by network links.

Management of each message queuing group is a relatively independent task. Therefore, it is best to assign each BEA MessageQ message queuing group a small set of application functions. Large or complex systems should be implemented as a network of queuing groups.

On large OpenVMS systems, the task of installing software is often assigned to a system manager who may not have detailed knowledge of each installed product. The system manager must learn the system resources required by each application, such as the size of the paging file, and amount of global memory and disk space required. OpenVMS `SYSGEN` parameters may have to be increased to accommodate installation of additional applications. To run BEA MessageQ and its applications, the system manager must configure the system with the appropriate resources to support the needs of all message queuing groups.

While some BEA MessageQ system parameters automatically adjust from default settings according to the load and available resources, the systems designer may have to make some decisions regarding BEA MessageQ resources and then set the parameters accordingly.

Queues need to be assigned to particular application services and various pools need to be sized. While this can be done in an iterative, ad-hoc way during the development stage, it is better to have a well-planned design and system model, and to make sizing decisions based on this model.

Design Paradigms

There are many design paradigms for distributed systems. No matter which paradigm is used, it should produce an abstract description that describes the information used by the system, and the things that happen in the system.

The abstract description is then transformed into a network of queues, application servers that read from the queues, and messages that flow between the application servers and encapsulate the characteristics and behavior. Each application server in a good design is assigned a specific function or a limited number of related functions to perform.

Traditional Functional Model

Using a traditional design methodology such as Yourdon/DeMarco¹, the system would be described by a series of data flow diagrams which show data flowing to and from abstract processes and data stores.

When using such a methodology, once the system data flow is known, the process of breaking down the diagram into physical processes and messages can start. Often a one-to-one association between physical process and abstract process can be made; for example, a process bubble in the diagram becomes a physical process. In some cases, several actions will be assigned to one physical process.

You should also consider the storage of data in media under application control, such as shared memory, disk files, or data base packages. The details of the choice for the type of storage are driven by access time requirements: how long the data must be stored, and how the data will be used.

1. See *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design* by Edward Yourdon and Larry L. Constantine, Prentice-Hall, Inc., 1979.

The decision to place data in some type of data store, necessitates the assignment of a physical process to manage writes or reads and writes to the data store. Data stores are often local to a particular computer or network node. The assignment of a server process to manage the store and a design that buffers access to the server by message queuing leads to a design with wide distributed access, fast and deterministic access times, and good scalability.

The location of physical processes on a network is sometimes well known because a data store managed by a service must be located on a particular node. In other cases, the decision to assign physical processes to network nodes is driven by load balancing considerations.

Object-Oriented Methodology

One of the difficulties that arise from a traditional design approach is that specification often becomes biased toward the flow of data rather than the understanding of the important underlying processes of the system. Object-oriented² analysis and design changes the focus from the data to the process, and produces a series of models in addition to the traditional functional model.

The end result is a series of abstract objects and methods. When object-oriented systems are implemented using BEA MessageQ, each major object is assigned to an application server process and a queue. The invocation of a method on an object corresponds to sending a message to an application server associated with the object.

At this stage in the design, there is enough information to assign physical processes to BEA MessageQ groups and to assign network node locations to those groups.

Determining Queue Sizes

In addition to the system flow, you need to determine expected arrival rates to key inputs of the system. One way to estimate this is to look at the expected or required response times to any particular action. So, if a input stream needs an response time of 0.5 seconds, you might expect that the arrival rate for that particular input might be 2

2. See *Object-Oriented Modeling and Design*, Rumbaugh, Blaha, Premerlani, Eddy, Lorensen, Prentice-Hall, Inc., 1991.

1 Preparing for BEA MessageQ Implementation

events per second. (The worst case then, as far as the system load, would be that the 2 events per second rate would be maintained over some relatively long period of time, say five minutes.)

The sum of input rates from all the events to a particular service determines a maximum input rate (and queue size) that the service must handle. If the events are correlated closely to messages, then the input messaging rates to the system can be determined.

The messaging rates for key inputs to the system, as well as the service rates for various processes, determine the queue size that the system needs to be configured to handle.

A rough way to do this is to use the following calculation³ for each queue:

- ◆ Assume for the purposes of calculation that arrivals occur randomly and independently of one another, with an average arrival rate that can be somehow determined or estimated.

Let,

- ◆ L = mean number of arrivals per time interval
- ◆ S = mean service time per arrival
- ◆ P = utilization = fraction of time service is busy = $L * S$
- ◆ Q = mean number of messages in the queue

Then for most systems, the value of q should be between the two following values:

$$Q_DET = P / (1 - P) - (P * P / 2 * (1 - P)) \quad \text{assumes constant service times}$$
$$Q_EXP = P / (1 - P) \quad \text{assumes random service times}$$

In many systems, you will need to consider the effect of server outages. Server outages occur when a server is unable to perform its function because it is explicitly shut down or because a key network link is down. When the server is down, the application must be able to take some type of action, such as storing the messages to be forwarded when the server becomes active again. Queue sizing analysis should also be done for the case when the server is down, and for when it is restarting. The queuing load during restart operations is often significantly larger than during normal operation.

3. You can use more sophisticated analytical formulas; for example, refer to *Queuing Systems, Volume 1: Theory* by Leonard Kleinrock, John Wiley & Sons, Inc., 1975.

In the case where the service and arrival rates are not known, the design and implementation work can still proceed by using the trial-and-error method. If there is not enough of a particular resource, add more and try again until the operation works.

Simulating Worst-Case Load Scenario

Another scheme to determine the resources that are required is to write input driver programs that simulate the worst-case load. Test drivers of this type are highly recommended, even when a complete systems design is available. Test drivers can be used in many phases of the project cycle, including design, modeling, implementation, and testing.

Servers can be also simulated, in a simple way, by programs coded as simple loops that read from an input queue, then pause for some interval before reading the next item. Using this methodology, you can build a high-level simulation that runs on the hardware and network (or a very similar network) on which the target application will run. Examining the resources used by the model can be very helpful in successfully sizing the production system.

Failover Provisions

Failover provisions must be taken into account early in the design process. Failover requires at least operational planning, and in most cases, requires application code to be written to support the failover process. If failover capability is required, it should be designed as an integral part of the application. It is difficult to back fit a failover process once an application is in production.

For example, recovery journals resides on physical media and this media must be accessible to both the primary node where the application is originally running and the backup node to which the application will fail over. In a VMSccluster, the media is easily accessible because access to disks can span network nodes. Outside of a VMSccluster, it is still possible to provide failover capability by physically moving or copying the media.

Design Summary

The following list summarizes basic questions you should consider when planning a distributed system that uses BEA MessageQ:

- ◆ Which applications must communicate with each other?
- ◆ Which computer systems do these applications run on?
- ◆ Where are the computer systems located?
- ◆ What networks and operating systems are these computer systems using?
- ◆ Where are users located?
- ◆ What backup/failover capability is needed?
- ◆ How will applications be developed and tested?
- ◆ What is an average transaction rate?
- ◆ What is the depth of backing storage?
- ◆ What are the service/arrival rates?
- ◆ What is the application data flow?
- ◆ What are failover provisions to back up the application?

2 Defining the Message Queuing Environment

To use BEA MessageQ, begin by configuring a message queuing group on your local system. For BEA MessageQ to exchange messages between networked systems, you must configure cross-group connections.

This chapter describes the initial configuration steps you must take to make BEA MessageQ operational at a minimum level. As you use BEA MessageQ and add new applications, you must upgrade the BEA MessageQ message queuing environment to satisfy new requirements. This chapter describes additional configuration and system management that are described in more detail in following sections.

This chapter discusses the following topics:

- Preparing to Define the BEA MessageQ Environment
- Creating a Message Queuing Group
- Starting and Stopping BEA MessageQ Groups
- Additional Configuration Tasks

Preparing to Define the BEA MessageQ Environment

Before defining your BEA MessageQ environment, you must create a system management account with the required privileges. Once the account has been created, you will need to perform some basic configuration tasks that will allow you to create, start, and stop message queuing groups.

BEA MessageQ Manager Account Privileges

Use the Authorize Utility (`AUTHORIZE`) to create a BEA MessageQ system management account with the following privileges:

ALTPRI	CMKRNL	DETACH	EXQUOTA
GROUP	GRPNAM	NETMBX	OPER
PSWAPM	SYSGBL	SYSLCK	SYSNAM
SYSPRV	TMPMBX	WORLD	

Basic Configuration Steps

After you install BEA MessageQ software using `SYS$UPDATE:VMSINSTAL`, you must perform several initial configuration steps. The executables and command procedures necessary to perform these can be found in:

```
disk:[DMQ$Vnn.EXE]
```

where *disk*: is the disk drive on which BEA MessageQ was installed and *vnn* is the version (for example, `V50`). After the initial configuration steps have been performed, the logical name `DMQ$EXE:` will point to this directory. Until then you need to specify the directory name.

The initial configuration steps are shown in Table 2-1.

Table 2-1 Initial Configuration Steps

Step	Function	Action
1	Run <code>DMQ\$CREATE_GROUP</code>	Creates the directory and initial configuration tables for a new BEA MessageQ group. This step is not required if you performed conversion to BEA MessageQ version 5.0 from an earlier version.
2	Run <code>DMQ\$STARTUP.COM</code> (without starting up servers)	Creates the required BEA MessageQ logical name table allowing group configuration.
3	Edit <code>DMQ\$INIT.TXT</code> file	Enters the configuration information (such as queue definitions, which servers to startup and inter-group message communications).
4	Run <code>DMQ\$STARTUP.COM</code>	Starts the BEA MessageQ group.
5	Run <code>DMQ\$SHUTDOWN.COM</code>	Shuts down a running BEA MessageQ group.

To maintain group configurations, you may need to perform the following tasks:

- Define connections to other message queuing groups in `DMQ$INIT.TXT` (see *Connecting to Other Message Queuing Groups*).
- Define and size message queues and global sections (see Chapter 4, “Configuring Message Queues and Global Memory”).
- Configure the Message Recovery Service (MRS) Server (see Chapter 5, “Configuring Message Recovery”).
- Configure the Selected Broadcasting Service (SBS) Server (see Chapter 6, “Setting Up Selective Broadcasting”).
- Initialize, define, and modify names that are available to processes at run time (see Chapter 7, “Creating Global Names”).

- Rebuild the language-specific include files and maintaining script symbols (see Chapter 7, “Creating Global Names”).
- Increase process quotas for BEA MessageQ servers in `DMQ$USER:DMQ$SET_SERVER_QUOTAS.COM` (see Chapter 11, “Sizing and Tuning the BEA MessageQ Environment”).

Creating a Message Queuing Group

The following topics are covered in this section:

- The BEA MessageQ Message Queuing Network
- Assigning Bus and Group IDs
- Creating Message Queuing Groups
- Default Directory Names
- Connecting to the BEA MessageQ Logical Name Table
- Editing `DMQ$INIT` to Configure a Group

Note: If you have upgraded an earlier version of BEA MessageQ to version 5.0 using the conversion utility (see the *BEA MessageQ Installation Guide for OpenVMS*), your messaging groups have already been created.

The BEA MessageQ Message Queuing Network

A BEA MessageQ message queuing network consists of one or more systems running BEA MessageQ. Each instance of BEA MessageQ running on a system is known as a BEA MessageQ group. Each group consists of queues, servers, and shared resources.

All queues within a group reside on the same system, share inter-process communications mechanisms (such as global memory, files and locks) and are handled by the same set of server processes. The servers, in turn, communicate to queues within their group, as well as to servers in other groups on the bus.

Before a group can start operation, the group's resources need to be configured. In BEA MessageQ for OpenVMS, each group has the following resources:

- A named logical name table
- A directory where configuration data is kept
- A directory where recovery journals are kept
- A directory where trace and error logs are kept
- A number of global memory sections, defined when the group is started
- A number of servers, each handling specific functions. These servers run as OpenVMS detached processes.

Assigning Bus and Group IDs

A message queuing group can consist of many message queues. A message queuing bus can have many message queuing groups. A networked system of computers may contain several message queuing buses which provide separate messaging domains.

The assignment of BEA MessageQ bus and group IDs should be done as part of the application system design. A well-planned design for bus and group IDs for the message queuing environment is similar to planning the topology of a network. If planned well, the network can handle changes as applications are added and changed. If planned poorly, it can be difficult to accomplish even minor changes and those changes can cause interruptions to network users.

In some situations, you may need to run more than one BEA MessageQ message queuing bus. A common situation for running multiple buses is when a test and production environment exist in the same network. BEA MessageQ allows multiple, concurrent BEA MessageQ buses to exist while ensuring that messages from different buses do not get mixed.

To create a bus, you create groups using the `DMQ$CREATE_GROUP.COM` command procedure and specify a new bus ID. The new bus will start running as soon as the groups within that message queuing bus are started. If more than one bus is required, the assignment of bus IDs should be designed using valid bus IDs in the range from 0001 to 9998.

Note: Bus IDs 0 and 9999 are reserved. We recommend that application developers and system designers control the assignment of bus and group IDs to applications.

An orderly assignment of group IDs may correlate group IDs to some functional, application, or organizational strategy. This strategy should allow the addition of future functional, application, or organizational areas.

Every group connected to a message queuing bus must have a unique group ID. BEA MessageQ allows up to 32000 group IDs per message queuing bus. A process for the registration and assignment of group IDs should ensure that each group ID is unique and does not conflict with other groups.

Creating Message Queuing Groups

Use the `DMQ$CREATE_GROUP.COM` command procedure to create message queuing groups. Execute `DMQ$CREATE_GROUP` once for each group to be created. This command procedure is located in the `[DMQ$V50.EXE]` directory.

`DMQ$CREATE_GROUP` has the following format:

```
$ DMQ$CREATE_GROUP bus_id group_id " " " " " " user_area log_area
```

where:

<code>bus_id</code>	Required; It Defines the message queuing bus domain and is an integer between 1 and 9998
<code>group_id</code>	Required; It defines a particular group within the message queuing bus and is an integer between 1 and 32000
<code>" "</code>	Not used.
<code>" "</code>	Not used.
<code>" "</code>	Not used.
<code>user_area</code>	Optional: Allows you to share other groups' configuration files and is an OpenVMS directory or logical name containing a directory (See "Sharing Group Configuration Files.")
<code>log_area</code>	Optional: It provides a directory location of log files and is an OpenVMS directory or logical name containing a directory

Default Directory Names

If you specify only the bus ID and group ID, DMQ\$CREATE_GROUP creates the following default directories, where *bbbb* is the 4-digit bus ID and *ggggg* is the 5-digit group ID:

- DMQ\$DISK:[DMQ\$V50.USER.*bbbb_ggggg*] --- for the user area
- DMQ\$DISK:[DMQ\$V50.LOG.*bbbb_ggggg*] --- for the log area
- DMQ\$DISK:[DMQ\$V50.MRS.*bbbb_ggggg*] --- for the MRS area

Listing 2-1 shows how a group (bus 15 and group 1) is created.

Listing 2-1 Invoking DMQ\$CREATE_GROUP

```
$ @DUAL:[DMQ$V50.EXE]DMQ$CREATE_GROUP 15 1
%CREATE-I-CREATED, DUAL:[DMQ$V50.USER.0015_00001] created
%CREATE-I-CREATED, DUAL:[DMQ$V50.LOG.0015_00001] created
%CREATE-I-CREATED, DUAL:[DMQ$V50.MRS.0015_00001] created
%DMQ-I-COPYFILE, Copying DMQ$BOOT.COM
%DMQ-I-COPYFILE, Copying DMQ$INIT.TXT
%DMQ-I-COPYFILE, Copying DMQ$SET_SERVER_LOGICALS.COM
%DMQ-I-COPYFILE, Copying DMQ$SET_SERVER_QUOTAS.COM
%DMQ-I-COPYFILE, Copying DMQ$TYPCLS.TXT
%DMQ-I-COPYFILE, Copying DMQCLSEC.TXT
%DMQ-I-COPYFILE, Copying FML32.H
%DMQ-I-COPYFILE, Copying HIDDEN_DATA_TYPES.PAS.
.
.
%DMQ-I-COPYFILE, Copying PAITYPBLI.TXT
%DMQ-I-COPYFILE, Copying PAITYPPAS.TXT
%DMQ-I-COPYFILE, Copying PAMS_ADA_ENTRY_POINT_.ADA
.
.
%DMQ-I-COPYFILE, Copying PAMS_ADA_TYPE_CLASS_.ADA
%DMQ-I-COPYFILE, Copying P_ENTRY.H
.
.
%DMQ-I-COPYFILE, Copying P_TYPECL.H
%DMQ-I-COPYFILE, Copying SYCUSTOMIZE.COM
%DMQ-I-COPYFILE, Copying TMENV.H
%DMQ-I-COPYFILE, Copying DMQ.TLB
%DMQ-I-COPYFILE, Copying DMQ.MLB
```

Connecting to the BEA MessageQ Logical Name Table

BEA MessageQ creates and utilizes a group logical name table for each BEA MessageQ group. The logical names contained in each table are used by BEA MessageQ to uniquely define the group.

The group logical name table is created when you run `DMQ$STARTUP.COM`. Once created, this logical name table will exist until either the system is rebooted or the `DMQ$DEASSIGN.COM` procedure is run.

Note: To create the BEA MessageQ group logical name table, execute `DMQ$STARTUP` *without* starting BEA MessageQ Servers. For a complete description of syntax for `DMQ$STARTUP`, see “Using `DMQ$STARTUP` to Invoke BEA MessageQ.” For an example of starting `DMQ$STARTUP` without starting servers, see “Invoking `DMQ$STARTUP` without Starting the Servers.”

BEA MessageQ places all the logical names that a group requires in this group logical name table. The table is named `DMQ$LNM_bbbb_ggggg`, where *bbbb* is the 4-digit bus ID and *ggggg* is the 5-digit group ID.

After the group logical name table is created, other processes (user or application) must run `DMQ$SET_LNM_TABLE.COM` to access the group table. A process must run this procedure in order to do the following:

- Attach to a message queuing group
- Execute BEA MessageQ command procedures or utilities
- Invoke BEA MessageQ main menu
- Compile, link, or run application programs

Note: We recommend that the `DMQ$SET_LNM_TABLE` command procedure be added to the user's `LOGIN.COM` file.

`DMQ$SET_LNM_TABLE` has the following format:

```
$ DMQ$SET_LNM_TABLE bus_id group_id [CREATE/UNSET/NOEXEC]
```

where:

<i>bus_id</i>	Required; Specifies the bus ID (an integer between 1-9998).
<i>group_id</i>	Required; Specifies the group ID (an integer between 1-32000)
CREATE/UNSET/NOEXEC	Optional; specifies alternate processing (a character string of CREATE, UNSET, or NOEXEC).
	If CREATE is used, the LNM table is created if it does not already exist.
	<p>If UNSET is used, the routine does the following:</p> <ul style="list-style-type: none"> ■ Fetches and removes the DMQ\$LNM_TBL logical name from the LNM\$FILE_DEV logical name list. ■ Removes DMQ\$LNM_TBL from the LNM\$PROCESS_DIRECTORY table. ■ Removes seven logicals from LNM\$PROCESS_TABLE: DMQ\$ENTRYRTL, DMQ\$EXECRTL, DMQ\$PSSRTL, DMQ\$VPS_USER, DMQ\$VPS_EXEC, DMQ\$LIBGP, DMQ\$LIBFML32.
	If NOEXEC is used, the routine will enter DMQ\$LNM_TBL in the LNM\$FILE_DEV logical name list and the LNM\$PROCESS_DIRECTORY table in supervisory mode rather than executive mode.

For example, if BEA MessageQ is installed on disk DUA1:, access the group with bus ID 15 and group ID 1 as follows:

```
$ @DUA1:[DMQ$V50.EXE]DMQ$SET_LNM_TABLE 15 1
%DMQ-S-SETLNM, Set to MessageQ LNM table DMQ$LNM_0015_00001
```

Editing DMQ\$INIT to Configure a Group

To create and maintain a BEA MessageQ group configuration, you need to perform the initial configuration, add new queues, maintain queue characteristics, maintain cross-group connections, maintain group tables, and modify group performance characteristics. Group configuration information is stored in the `DMQ$INIT.TXT` file which is located in the `DMQ$USER` directory.

The `DMQ$INIT.TXT` file contains information about:

- Group configuration parameters
- Which servers the group will run and their characteristics
- Which other groups the local group must communicate with
- What queues are defined on this group and their characteristics
- Group performance characteristics

Edit the `DMQ$INIT.TXT` file to add and change group configuration information. This file should be maintained by the system manager or other individual responsible for maintaining and managing the BEA MessageQ environment. The file can be edited by any of the standard system provided editors or through the `DMQ$MENU` utility procedure (refer to Chapter 9, “BEA MessageQ Main Menu and Utilities,” for more information on `DMQ$MENU`).

The `DMQ$INIT.TXT` file contains the following sections that you must customize for each group. Each section begins with a percent sign (%) and the name or an abbreviation for the name of the section. The following characters are treated as comment characters in the `DMQ$INIT.TXT` file: exclamation point (!), pound sign (#), semicolon (;) and asterisk (*). Table 2-2 describes each section of `DMQ$INIT.TXT` and tells you where to find more information about how to configure each section of the file.

Table 2-2 Sections of the DMQ\$INIT.TXT File

Section Header	Name	Description
%PROFILE	Profile	Must precede all other sections and specifies global or COM Server-specific parameters (See “Setting Servers and Configuring Groups in the Profile Section”).
%XGROUP	Cross-group Communications	Provides the link driver with the information it needs to connect to other message queuing groups. The Cross-group section should contain a entry for the node on which the local group is running. (See Chapter 3, “Configuring Cross-group Connections”).
%ROUTE	Message Routing	Enables cross-group communications paths among nodes with no direct network links (See Chapter 3, “Configuring Cross-group Connections”).
%CLS	Client Library Server	Enables the support of multiple BEA MessageQ Client connections to a single OpenVMS process using either DECnet and TCP/IP transports (See Chapter 8, “Configuring the BEA MessageQ Client Library Server”).
%BUFFER	Buffer Pool Configuration Table	Controls the size of the buffer pool where messages are stored (See Chapter 4, “Configuring Message Queues and Global Memory.”)
%QCT	Queue Configuration Table	Contains the specifications for all permanent queues. The default QCT section requires tailoring to meet your application needs. (See Chapter 4, “Configuring Message Queues and Global Memory.”)
%SBS	Selective Broadcast Services (SBS)	Controls the operation of the local SBS Server (See Chapter 6, “Setting Up Selective Broadcasting.”)
%MRS	Message Recovery System (MRS)	Controls the operation of the local MRS Server (See Chapter 5, “Configuring Message Recovery.”)
%GNT	Group Name Table	Contains alias and bind queue names and their addresses (See Chapter 5, “Configuring Message Recovery.”)
%NAM	Naming Agent	Contains the group numbers in which the Naming Agent will run. (See Chapter 7, “Creating Global Names.”)

Setting Servers and Configuring Groups in the Profile Section

A BEA MessageQ group includes a number of server processes. Each server process is dedicated to a specific function or set of functions. The main overseer of each BEA MessageQ message queuing group is the communications (COM) server. During startup, the COM Server process starts up subsidiary servers depending upon the contents of the Profile section. You can start the following servers by indicating YES for each server entry in `DMQ$INIT.TXT` file:

- MRS SERVER—handles recoverable messaging
- SBS SERVER—handles message broadcasting and AVAIL/UNAVAIL notification
- JRN SERVER—handles the Dead Letter and Post Confirmation Journals
- QTRANSFER SERVER—handles failing over of recoverable message queues

To request these BEA MessageQ Servers to start up, set the appropriate `ENABLE_xxx` parameters to YES in the profile section of `DMQ$INIT.TXT`. Listing 2-2 shows a sample Profile section. It specifies information that is global to the entire group or information that is needed by the COM server.

Listing 2-2 Sample Profile Parameters

```
%PROFILE          ***** Profile Parameters *****
*
ACCEPT_KILL_CMD      NO          ! Control MONITOR terminate requests
ENABLE_XGROUP        NO          ! Enable MessageQ cross-group access
XGROUP_VERIFY        NO          ! Limit incoming cross-group connections
ENABLE_MRS           YES         ! Enable MessageQ Message Recovery Services
ENABLE_JRN           NO          ! Enable MessageQ Message Journaling Services
ENABLE_SBS           YES         ! Enable MessageQ Selective Broadcast Services
ENABLE_QXFER         NO          ! Enable MessageQ MRS Queue Transfer Services
FIRST_TEMP_QUEUE     200         ! Select start of temp queue pool (200-3950)
XGROUP_TABLE_SIZE    25          ! Select max number of group entries (25-32000)
RCV_MSG_QUOTA_METHOD MAX        ! Select rcv msg quota deduction (MIN | MAX)
ATTACH_TMO           600         ! Select PAMS_ATTACH_Q timeout (100-36000)
GROUP_MAX_USER_QUEUE 999         ! Maximum temp queue number (999-3999)
GROUP_MAX_MESSAGE_SIZE 128000    ! Maximum number of bytes per message
*DEFAULT_NAMESPACE_PATH /DMQNS/ ! Default namespace path for the Naming Agent
*
%EOS
```

Profile Section Parameters

Table 2-3 shows the parameters in the Profile section.

Table 2-3 Profile Parameters

Parameter	Range	Default	Description
ACCEPT_KILL_CMD	YES/NO	NO	Controls COM Server termination requests from the DMQ\$MONITOR utility. A setting of NO means that any message requests to shut down COM Server are ignored.
ENABLE_XGROUP	YES/NO	NO	Enables BEA MessageQ DECnet or TCP/IP network access. A setting of YES causes the appropriate link drivers to start up to perform cross-group communications.
XGROUP_VERIFY	YES/NO	NO	Validates incoming DECnet or TCP/IP connections. A setting of YES causes all connections to be validated against the cross-group connection table, much like an access control list (ACL).
ENABLE_MRS	YES/NO	YES	Controls the enabling or disabling of BEA MessageQ Message Recovery Services. A setting of YES enables the MRS Server to be started which is necessary for doing guaranteed message delivery via Store and Forward (SAF) and/or Destination Queue Files (DQF).
ENABLE_JRN	YES/NO	NO	Controls the enabling or disabling of BEA MessageQ Message Journaling Services. A setting of YES enables the JRN Server to be started which is necessary for Post Confirmation Journaling (PCJ) and/or Dead-Letter Journaling (DLJ).
ENABLE_SBS	YES/NO	YES	Controls the enabling or disabling of BEA MessageQ Selective Broadcast Services and/or AVAIL Services.

2 Defining the Message Queuing Environment

Table 2-3 Profile Parameters

Parameter	Range	Default	Description
ENABLE_QXFER	YES/NO	NO	Controls the enabling or disabling of BEA MessageQ MRS Queue Transfer Services.
FIRST_TEMP_QUEUE	200—3950	200	Selects start of the temporary queue pool. This defines the size of the permanent and temporary queue pools for a group.
XGROUP_TABLE_SIZE	25—32000	25	Selects maximum number of group entries. This defines the maximum number of concurrently active groups tracked by the COM Server. This does not affect the maximum group number.
RCV_MSG_QUOTA_METHOD	MIN/MAX	MAX	Selects type of receive byte quota deductions. The two methods are: MIN, which deducts only the number of bytes used from the receive byte quota. MAX, which deducts the size of the buffer used to hold the message (LARGE, MEDIUM, or SMALL) as defined in the %BUFFER section.
ATTACH_TMO	100—36000	600	Selects the maximum elapsed time that an attach or locate queue operation can take before a status of PAMS__TIMEOUT is returned to the caller. This number is in units of tenths of a second.
GROUP_MAX_USER_QUEUE	999—3999	999	Sets the highest user queue number
GROUP_MAX_MESSAGE_SIZE	2048—4194304	128000	Sets the maximum number of bytes per message.
DEFAULT_NAMESPACE_PATH	—	/DMQNS/	Sets the default namespace path for the Naming Agent. The VMS root for this path is defined in DMQ\$BOOT.COM with the DMQNS_DEVICE logical name. This parameter is commented out by in the DMQ\$INIT.TXT template. See “Default Namespace Path Definition” for more information.

Default Namespace Path Definition

The `DMQNS_DEFAULTPATH` logical name is now supported, and is defined in `DMQ$BOOT.COM`. Using the `DMQNS_DEFAULTPATH` logical name allows you to remove the `DEFAULT_NAMESPACE_PATH` definition from the `DMQ$INIT.TXT` file, yet still define the path information needed by the Naming Agent Server. See Chapter 7, “Creating Global Names,” for more information on defining a name space.

Setting Network Connections in the Cross-group Section

To establish connections to other BEA MessageQ groups, you must modify the cross-group connection table of the `DMQ$INIT.TXT` file. This table provides the COM Server with the information required to connect to other BEA MessageQ groups through DECnet or TCP/IP connections.

One entry for each BEA MessageQ group should be in the `Xgroup` section if the group in question is sending messages to another.

Listing 2-3 is a sample of the cross-group connection (`%XGROUP`) table. Refer to Chapter 3, “Configuring Cross-group Connections” for more information on how to establish cross-group connections. A group must have an entry in the cross-group table for itself.

Listing 2-3 Cross-group Connection Table

```

%XGROUP      ***** Cross-Group Connection Table *****
*
*
*          gen buf   buf   recon   dly win  transport
*          cnt warn pool   sec     sec msg  type  endpt
*-----
*GR1_TCP      1  ipaddr1.company.com   Y  -1  175   -1  -1  -1  TCPIP  2001
*GR1_TCP      1  ipaddr2.company.com   Y  -1  175   -1  -1  -1  TCPIP  2001
*GR1_DECNET   1  daddr1                      Y  -1  -1    -1  -1  -1  DECNET
*GR1_DECNET   1  daddr4                      Y  -1  -1    -1  -1  -1  DECNET
*
*GR2_TCP      2  ipaddr3.company.com   Y  -1  175   -1  -1  -1  TCPIP  2002
*GR1_TCP      1  ipaddr3.company.com   Y  -1  175   -1  -1  -1  TCPIP  2002
*GR2_DECNET   2  daddr4                      Y  -1  -1    -1  -1  -1  DECNET
*GR2_FAILOVER 2  daddr5                      Y  -1  -1    -1  -1  -1  DECNET
*
%EOS

```

Configuring Client Library Server in the CLS section

Use the Client Library Server configuration table to specify a PC-based client server endpoint (address), transport type, and maximum number of clients allowed for each server.

Listing 2-4 is a sample of the Client Library Server configuration table. Refer to Chapter 8, “Configuring the BEA MessageQ Client Library Server” for more information on how to configure the CLS section.

Listing 2-4 Client Library Server Configuration Table

```
%CLS      **** Client Library Server Configuration Table ****
*
*
* Endpoint      Transport      Maximum #      Security
* -----      -
* 5000          TCPIP          16             dmq$user:cl_5000.sf
* 5001          TCPIP          16
* 6000          DECNET         32
*
*
*EOS
*
```

Configuring Local Buffer Pools in the Buffer section

BEA MessageQ uses sections of global memory to process messages. The section of global memory used for any given message is determined by the size of the message. There are global sections for small, medium, and large messages. The buffer pool configuration table is where the sizes of messages, the number of messages and threshold parameters for each global sections are defined.

Listing 2-5 is a sample of the buffer pool configuration table. Refer to Chapter 4, “Configuring Message Queues and Global Memory” for more information on how to change BEA MessageQ memory allocation.

Listing 2-5 Buffer Pool Configuration Table

```

%BUFFER          ***** Buffer Pool Configuration Table *****
*
*Msg-Block-Type  Byte-Size      Number      Warning-level  Reserve
*-----*-----*-----*-----*-----*
SMALL             256                50           10              2
MEDIUM           5000               10            2              1
LARGE            64000              3             1              0
%EOS
*
    
```

Defining Queues in the Queue Configuration Section

The fundamental entity within a message queuing system is the queue. Each message queue has a number of definable attributes. As your BEA MessageQ environment matures, new queues will be added to groups and existing queue attributes will be changed.

The message queue characteristics that you can define are the queue name, queue number, performance attributes, queue type, owner of the queue, and the life span of the queue. Queue configuration information is stored in the Queue Configuration Table of the `DMQ$INIT.TXT` file.

Listing 2-6 is a sample of a small portion of the queue configuration table. Refer to Chapter 4, “Configuring Message Queues and Global Memory” for detailed information on how to define and configure queues.

Listing 2-6 Queue Configuration Table

```

%QCT          ***** Queue Configuration Table *****
*
*          ---Pool Quota---  UCB  Q  Q  Confrm Perm Name  Chk
*  Queue Name      Num  Bytes Msgs Ctrl Send Type Own  Style Act  Scope ACL
*-----*-----*-----*-----*-----*-----*-----*
*  Sample Queues
QUEUE_1           1   64000  100 All    .    .    .    II  Y    L    N
QUEUE_2           2   64000  100 Msg    .    .    .    EI  Y    L    N
QUEUE_3           3   64000  100 Byte  .    .    .    EO  Y    L    N
QUEUE_4           4   64000  100 None  .    .    .    .  Y    L    N
QUEUE_5           5   64000  100    .    .    .    .  Y    L    N
    
```

2 Defining the Message Queuing Environment

```
QUEUE_6          6  64000  100  .  .  .  .  .  .  Y  L  N
QUEUE_7          7  64000  100  .  .  .  .  .  .  Y  L  N
QUEUE_8          8  64000  100  .  .  .  .  .  .  Y  L  N
QUEUE_9          9  64000  100  .  .  .  .  .  .  Y  L  N
QUEUE_10         10 64000  100  .  .  .  .  .  .  Y  L  N
```

*

%EOS

Setting Broadcast System Parameters in the SBS Section

The Selective Broadcast Server provides the function of broadcasting messages to interested parties. To set broadcast system parameters, edit the Selected Broadcast Services (SBS) Server section of the `DMQ$INIT.TXT` file.

Listing 2-7 is a sample of the SBS Server section. Refer to Chapter 6, “Setting Up Selective Broadcasting” for more detailed information on configuring message broadcasting.

Listing 2-7 SBS Server Initialization Section

```
%SBS  ***** SBS Server Initialization Section *****
*
*   NOTE: Heartbeat interval is in units of 1 millisecond
*
HEARTBEAT          2000
*
*           ---- Service ----
*           ID  Prot/Xport
COMM_SERVICE      10  DG/DMQ      ! default emulated broadcast path
  GROUPS *
  REGISTER *
END_COMM_SERVICE
*
%EOS
```

Setting Message Recovery System Parameters in the MRS Section

The Message Recovery System has several configurable attributes. To set recovery system parameters, edit the Message Recovery Service (MRS) Server section of the `DMQ$INIT.TXT` file. This section allows you to configure the mechanism for reliable message recovery and delivery.

The MRS file system is split up into a variable number of fixed-sized areas. MRS manages the creation, assignment, and deletion of files without user intervention. The total disk space taken up by incoming (DQF) and outgoing (SAF) journal files can be controlled by setting three parameters:

- `AREA_SIZE`
- `NUM_DQF_AREAS`
- `NUM_SAF_AREAS`

Listing 2-8 is a sample of the MRS Server initialization section. Refer to Chapter 5, “Configuring Message Recovery” for more information on how to configure the message recovery system.

Listing 2-8 MRS Server Initialization Section

```
%MRS      ***** MRS/JRN Server Initialization Section *****
*
AREA_SIZE          512 ! disk blks per file (min:128, max:16384, def:512)
NUM_DQF_AREAS     1000 ! min:100, max:1000000, default:1000
NUM_SAF_AREAS     1000 ! min:0, max:1000000, default:1000
NUM_PCJ_AREAS     1000 ! min:0, max:1000000, default:1000
NUM_DLJ_AREAS     1000 ! min:0, max:1000000, default:1000
NUM_MESSAGES      512 ! min:128, max:2147483647, default:512
NUM_QUEUES        128 ! min:128, max:2147483647, default:128
CACHE_PERCENTAGE  90 ! % rcv quota for MRS msgs (min:1, max:100, def:90)
USE_HIGH_WATER_MARK YES ! checkpt MRS sizing params to disk (YES/NO)
LOAD_MRS_CTRS     YES ! init recoverable msg ctrs on startup (YES/NO)
RCVR_ONLY_CONFIRM YES ! limit msg confirms to receiving process (YES/NO)
XGRP_JRN_CTRL     NO ! allow JRN cntrl msgs from other groups (YES/NO)
REDELIVERY_TIMER  10 ! integer seconds (min:0, max:5000, default:10)
*
PCJ_FILENAME      DMQ$MRS:MRS_%bg.PCJ ! char[64] - %bg is a macro that
DLJ_FILENAME      DMQ$MRS:MRS_%bg.DLJ ! char[64] - expands to bus_group
```

Assigning the Groups in which the Naming Agent Will Run

Listing 2-9 is a sample of the section of the group initialization file used to configure the groups in which the Naming Agent will run. Refer to Chapter 7, “Creating Global Names,” for more complete information on how to configure BEA MessageQ to support its global naming feature.

Listing 2-9 Naming Agent Initialization Section

```
%NAM ***** Naming Agent Section *****
* This section consists of a maximum of 2 entries consisting of
* a keyword, "NA_GROUP", followed by the group number of a group
* where a naming agent is running

*NA_GROUP 1
NA_GROUP 2
%EOS
```

Starting and Stopping BEA MessageQ Groups

The following topics are covered in this section:

- Using DMQ\$STARTUP to Invoke BEA MessageQ
- Starting Groups Under an Older Version of BEA MessageQ
- Starting Application Programs in a Message Queuing Group
- Application Startup Process
- Two Methods for Connecting
- Shutting Down a Running BEA MessageQ Group

Using DMQ\$STARTUP to Invoke BEA MessageQ

To set up the BEA MessageQ environment and begin using the BEA MessageQ software you need to run the DMQ\$STARTUP command procedure. You can execute DMQ\$STARTUP interactively or during system startup by adding a call to the SYS\$STARTUP_VMS command.

The DMQ\$STARTUP command procedure automatically sets the named logical name table for the current process by invoking the DMQ\$SET_LNM_TABLE command procedure. After the BEA MessageQ application is started, you can access its main menu and use the BEA MessageQ main menu to compile, link, and run the utilities.

DMQ\$STARTUP has the following format:

```
$ @DMQ$STARTUP bus_id group_id timeout start_servers " " user_area log_area
```

where:

<i>bus_id</i>	Required. It specifies the bus ID of the bus you want to start and is an integer between 1 and 9998.
<i>group_id</i>	Required. It specifies the group ID of the group you want to start and is an integer between 1 and 32,000.
<i>timeout</i>	Optional. It specifies the server startup timeout in seconds.
<i>start_servers</i>	Optional. It specifies whether you want to start the BEAMessageQ servers for this group and is either Y or N. The default is Y to start the servers. If you do not want to start the servers, specify N for this parameter.
" "	Not used.
<i>user_area</i>	Optional. It specifies the directory name for storing user files; is a directory specification or logical name containing the directory specification. The logical name DMQ\$USER points to this location which contains the group-specific DMQ\$BOOT.COM and DMQ\$INIT.TXT files.
<i>log_area</i>	Optional. It specifies the directory name for storing log files and is a directory specification or logical name containing the directory specification. The logical name DMQ\$LOG points to this location.

To use the BEA MessageQ default directories, specify only the bus and group ID parameters. Specify the parameters for user area or log area when starting a group that uses configuration files that are different from the default.

Invoking DMQ\$STARTUP without Starting the Servers

Sometimes, you may want to configure a BEA MessageQ message queuing group without starting it; for example, during initial group configuration. To define the BEA MessageQ environment, run the DMQ\$STARTUP.COM procedure and specify the start *start_servers* parameter as N to prevent the servers from starting.

Listing 2-10 shows how to invoke DMQ\$STARTUP without starting the servers. It defines the environment for bus 99, group 50, but does not start the servers. After you run DMQ\$STARTUP, you can customize the messaging group using the DMQ\$INIT.TXT file.

Listing 2-10 Invoking DMQ\$STARTUP without Starting Servers

```
$ @DUAL:[DMQ$V50.EXE]DMQ$STARTUP 99 50 " " N
DMQ$STARTUP.COM - 12-NOV-1999 14:40:46.88

    Bus: 0099
    Group: 00050
        Disk: DISK$DUAL:
        Timeout:
    Start Servers: No
    DMQ Version: DMQ$V50
    User Area: DMQ$DISK:[DMQ$V50.USER.0099_00050]
    Log Area: DMQ$DISK:[DMQ$V50.LOG.0099_00050]

%DQ-S-SETLNM, Set to MessageQ LNM table DMQ$LNM_0099_00050

...Installing DMQ$MSGSHR
...Installing DMQ$EXECRTL
...Installing DMQ$ENTRYRTL
...Installing DMQ$VPS_EXEC
...Installing DMQ$SET_LNM
```

Note: BEA MessageQ places all the logical names that a group requires in the group-specific logical name table, DMQ\$LNM_ddd_ggggg, where *ddd* is the 4-digit bus ID and *ggggg* is the 5-digit group ID.

Starting Groups Under an Older Version of BEA MessageQ

BEA MessageQ for OpenVMS is designed to allow all versions to coexist on the same system and the same disk. Each version has its own startup command procedure called `DMQ$STARTUP.COM`. For example, to start a version 4.0 group (bus 20, group 36), execute the following:

```
$ @DUA0:[DMQ$V40.EXE]DMQ$STARTUP 20 36
```

Note: When running multiple versions of BEA MessageQ in the same message queuing environment, do not define two groups with the same bus and group IDs if they are to run on the same system simultaneously.

Starting Application Programs in a Message Queuing Group

Application programs using BEA MessageQ for OpenVMS are usually written as a number of detached cooperating server processes within a single message queuing group. Each server process handles specific functions and communicates with clients and other servers via BEA MessageQ messages.

Application Startup Process

The application startup procedure starts the message queuing group along with the BEA MessageQ Servers. It then detaches application server processes with the DCL command `RUN/DETACH`.

Two Methods for Connecting

Because BEA MessageQ does not use `GROUP` or `SYSTEM` logical name tables, applications that are started by the `RUN/DETACH` command need a mechanism to attach the detached process to the named logical name table associated with a given bus ID and group ID.

There are two ways for a detached process to connect to a BEA MessageQ bus and group:

- Method #1: Detach the process with DCL context.
- Method #2: Does not require DCL context, but does require a unique association between a message queuing group and a UIC code.

Detaching a Process with DCL Context

The detached process is given a DCL command stream that sets the BEA MessageQ logical name table and runs the application program image. BEA MessageQ uses this mechanism when it detaches its servers. Use the `DMQ$PROCESS_START` command procedure as a starting point in running the detach process with DCL context.

`DMQ$PROCESS_START` builds a temporary command procedure that is passed to `LOGINOUT.EXE` by the `/INPUT` qualifier. The temporary command procedure is a single-line invocation of `DMQ$DETACH_PROCESS` with the required parameters filled in.

Use the following command to run the `DMQ$PROCESS_START` command procedure:

```
$ @DMQ$EXE:DMQ$PROCESS_START executable_name process_name priority page_file
```

where:

<i>executable_name</i>	Required; Specifies the name of the executable that you want to start.
<i>process_name</i>	Required; Specifies the name of the process that you want to start.
<i>priority</i>	Optional; Allows you to specify a process priority.
<i>page_file</i>	Optional; Allows you to specify the size of the page file.

Detaching a Process without DCL Context

You can detach an application process without a DCL context by making a unique association between a message queuing group and a UIC code. This allows the detached process to access the group logical name table to pick up its configuration and startup information. Use `DMQ$COPY_LNM_TABLE` to copy the logical names from the BEA MessageQ LNM table to either a group or system LNM table.

After the logical names have been copied, use the DCL command `RUN/UIC` to detach a user application process directly. This approach is recommended for detaching processes in environments where a UIC can be associated easily and uniquely with each message queuing group.

Listing 2-11 shows a detached process attached to bus 15, group 1.

Listing 2-11 Accessing the Group Logical Name Table

```
#!/In this DCL fragment, the program SYS$LOGIN:MYPROG.EXE is started
#!/as a detached process connected to bus:15 group:1
#!/DMQ$COPY_LNM_TABLE is used to copy the MessageQ logical name
#!/table into the group table associated with the current UIC.
$
$ myarea          = f$strnlm("SYS$LOGIN")
$ proc_name       := MYPROG
$ log_file        := 'myarea'MYPROG.LOG
$ err_file        := 'myarea'MYPROG.ERR
$ bus_id          := 00015
$ grp_id          := 00001
$
$ !Connect to MessageQ
Logical Name Table
$ @DMQ$EXE:DMQ$SET_LNM_TABLE 'bus_id' 'grp_id'
$
$ !Copy the MessageQ
Logical Name Table into the current group table
$ @DMQ$EXE:DMQ_COPY_LNM_TABLE GROUP
$
$ run /detached SYS$LOGIN:MYPROG.EXE      -
    /priv   = (noall,netmbx,tmpmbx)      -
    /output = 'log_file'                  -
    /error  = 'err_file'                  -
    /name   = 'proc_name'
```

2 Defining the Message Queuing Environment

The `DMQ$COPY_LNM_TABLE` command procedure copies logical names from the named logical name table that is currently connected to the `GROUP` or `SYSTEM` logical name table.

The `DMQ$COPY_LNM_TABLE` command procedure has the following format:

```
$ DMQ$COPY_LNM_TABLE system_type octal_group_UIC
```

where:

<i>system_type</i>	Required; specifies the system type (<code>GROUP</code> or <code>SYSTEM</code>) of the group you want to start
<i>octal_group_UIC</i>	Optional; specifies the octal group UIC number

Shutting Down a Running BEA MessageQ Group

Use the `DMQ$SHUTDOWN.COM` procedure to shut down a running group. You can run the command procedure directly or you can run it through option 2 of the `DMQ$MENU.COM` utility procedure.

The `DMQ$SHUTDOWN` command procedure forces the COM Server process to exit either gracefully or instantly (known as `stop_fast`). Once the COM server exits the rest of the group server processes will also shutdown. The procedure can optionally deinstall the run-time libraries (RTLs).

`DMQ$SHUTDOWN` has the following format:

```
$ @DMQ$EXE:DMQ$SHUTDOWN bus_id group_id rtl_deinstall stop_fast
```

where:

<i>bus_id</i>	Required; specifies the bus ID of the RTL's bus.
<i>group_id</i>	Required; specifies the group ID of the RTL's group.
<i>rtl_deinstall</i>	If set to <code>YES</code> , deinstalls all the installed BEA MessageQ images and RTLs for that group. If set to <code>NO</code> (the default), does not deinstall RTLs.

stop_fast Default is YES, which uses the FORCEX utility to stop the COM server and bring down all other BEA MessageQ processes. Specifying NO will result in a graceful shutdown. During a graceful shutdown BEA MessageQ waits until all queues have been drained before shutting down.

The following example shows the COM Server shutdown command for bus 15 and group 1.

```
$ @DMQ$EXE:DMQ$SHUTDOWN 15 1
```

Note: We recommend that you use DMQ\$SHUTDOWN before you install new patches to the BEA MessageQ kit or when using the SYS\$SHUTDOWN procedure. Repeat DMQ\$SHUTDOWN to shut down each running COM Server on the system to which you need to apply the software patch.

Use the main menu to restart the BEA MessageQ Servers.

Additional Configuration Tasks

The following topics are covered in this section:

- Configuring Event Logging
- Adding Queue Names to Network-wide Namespace
- Changing Parameters in the Running Group
- Deleting Groups
- Defining Message Type and Class Codes
- BEA MessageQ Hints and Tips

Configuring Event Logging

BEA MessageQ offers a variety of options for capturing and viewing system events. Event logging captures important information regarding the success and failure of BEA MessageQ operations. You can use the information in BEA MessageQ event logs to troubleshoot system problems. BEA MessageQ allows you to configure a variety of event logging options, including the ability to:

- Direct and redirect output streams
- Switch specific types of stream output on and off
- Manipulate and view log files to monitor BEA MessageQ output and isolate operational problems

If you are having problems with a BEA MessageQ Server, look at the log files for that server. Refer to Chapter 12, “Managing a BEA MessageQ Environment,” for more information on how to locate and view server log files. If you are having difficulty starting the BEA MessageQ group, look at the COM Server log file. This may include informative messages if the COM Server is having difficulty during startup.

Adding Queue Names to Network-wide Namespace

BEA MessageQ includes a capability for creating and maintaining a global name space and for providing runtime look up of name-to-queue address translations. These services are provided by the BEA MessageQ Naming Agent.

In addition to using the BEA MessageQ global naming capability, you can continue to use Distributed Namespace names using the `NA` option of the `DMQ$MGR_UTILITY` utility. Refer to Chapter 7, “Creating Global Names,” for information on configuring and using the BEA MessageQ global naming feature. Refer to Chapter 10, “Using BEA MessageQ System Management Utilities,” for information on using the BEA MessageQ Manager utility.

Changing Parameters in the Running Group

BEA MessageQ allows you to change some group characteristics at run-time. To do so, you must edit the `DMQ$INIT.TXT` file for the group and then run the `DMQ$LOADER` program to modify queues, links, names, and other modifiable group characteristics. The ability to modify some group characteristics is dependent upon the state of the object being modified.

Group initialization file parameters that are removed from the `DMQ$INIT.TXT` file will not be removed from the running group. To remove characteristics, you must stop the group and then restart it after modifying the group initialization file to delete the unwanted characteristics.

BEA MessageQ allows a limited number of multireader queues to be added to a running group without restarting the group. The limit is twice the initial number of MRQ entries or 20, whichever is greater. In addition, links to a remote group can be added to a group initialization file and activated by running the `DMQ$LOADER` program.

You can run the `DMQ$LOADER` from the BEA MessageQ main menu by selecting Option 10. `DMQ$LOADER` also runs during the servers startup (`@DMQ$STARTUP`). Table 2-4 describes the parameters in the group initialization file that can be modified at runtime.

Table 2-4 Modifiable Parameters in the Group Initialization File

Init File Section	Parameter	Run-time Restriction?
%PROFILE	ACCEPT_KILL_COMMAND	No
	ATTACH_TMO	No
	DEFAULT_NAMESPACE_PATH	No
%CLS	MAX_CLIENTS	Yes. CLS must be stopped. This parameter applies only to OpenVMS systems.
	SECURITY_FILE	Yes. CLS must be stopped.

2 Defining the Message Queuing Environment

Table 2-4 Modifiable Parameters in the Group Initialization File

Init File Section	Parameter	Run-time Restriction?
%XGROUP	RECONNECT	Yes. The link must be disabled.
	RECONN_TIMER	Yes. The link must be disabled.
	WINDOW_DELAY	Yes. The link must be disabled.
	WINDOW_SIZE	No
%QCT	BYTE_QUOTA	No
	MESSAGE_QUOTA	No
	MESSAGE_QUOTA_ENABLE	No
	BYTE_QUOTA_ENABLED	No
	TYPE	Yes. The queue must be empty and have no processes attached. When changing a primary queue to a secondary queue, the primary queue cannot have any secondary queues defined.
	OWNER	Yes. The queue must be empty and have no processes attached. To set this parameter to a value other than zero, the queue must be defined as a secondary queue, and the owning queue must be defined and be a primary queue.
	MRS_CONFIRM_STYLE	No
PERM_ACTIVE	PERM_ACTIVE	Yes. The queue must be empty and have no processes attached.
	SECURITY_ENABLED	Yes. No change may be made to the security field unless the queue is empty and it has no attached processes.

Deleting Groups

Use the following command to run the `DMQ$DELETE_GROUP` command procedure:

```
$ @DMQ$EXE:DMQ$DELETE_GROUP bus_id group_id logging_flag
```

where:

<i>bus_id</i>	Required. Specifies the bus ID for the group to be deleted.
<i>group_id</i>	Required. Specifies the group ID for the group to be deleted.
<i>logging_flag</i>	Optional. Set to <code>Y</code> to enable logging or <code>N</code> to disable logging for the deletion. The default value is <code>Y</code> .

To delete a message queuing group, take the following steps:

Step	Action
1. Bring down the group's servers	Run <code>DMQ\$SHUTDOWN.COM</code> and specify the bus ID, group ID, and <code>Y</code> to remove installed images.
2. Delete the group	<p>Run <code>DMQ\$DELETE_GROUP.COM</code> specifying the bus and group ID of the group to delete. If the group shares directories with another group, do NOT run <code>DMQ\$DELETE_GROUP.COM</code>. Instead, delete the files specific to that group (see Sharing Group Configuration Files). These files are:</p> <ul style="list-style-type: none"> ■ <code>DMQ\$bbbbggggg*.DAT</code> in the <code>DMQ\$ACCESS</code> directory (see Chapter 13, "BEA MessageQ Security") ■ <code>DMQ\$CHKPT_bbbb_ggggg.DAT</code> ■ <code>DMQ\$*_bbbb_ggggg.LOG</code> in the <code>DMQ\$LOG</code> directory ■ <code>DMQ\$MRS_bbbb_ggggg_*. *</code> in the <code>DMQ\$MRS</code> directory (see Chapter 5, "Configuring Message Recovery")

Defining Message Type and Class Codes

BEA MessageQ software customization includes defining the symbolic usage of type and class codes within BEA MessageQ application programs and the BEA MessageQ script facility. The `DMQ$TYPCLS.TXT` file contains symbolic definitions of type and class codes. Refer to Chapter 7, “Creating Global Names” for more information on how to edit this file to create or change application-specific type and class codes.

BEA MessageQ Hints and Tips

This section provides some helpful hints for configuring your BEA MessageQ environment, including:

- Modifying Your Default Editor
- Defining BEA MessageQ Symbols
- Startup Synchronization
- Redirecting Configuration and Log Files

Modifying Your Default Editor

The default editor used by BEA MessageQ command procedures is `DECTPU`. To change the default editor, change the global symbol `DMQ$$EDITOR`. For example, enter the following command to change the default editor to invoke your own customized `DECTPU` section file:

```
$ DMQ$$EDITOR ::= EDIT/TPU/SECTION=SYS$LOGIN:MY_SEC.TPU$SECTION
```

Defining BEA MessageQ Symbols

You can define symbols that execute the command procedures commonly used by BEA MessageQ application developers. We recommend that you define these symbols in the `SYS$MANAGER:SYLOGIN` command procedure.

Define the following symbol for setting the logical name table:

```
$ DMQ_SET ::= @DUA1:[DMQ$V50.EXE]DMQ$SET_LNM_TABLE
```

Define the following symbol for invoking the BEA MessageQ main menu:

```
$ DMQ ::= @DMQ$EXE:DMQ$MENU
```

Startup Synchronization

The `DMQ$STARTUP.COM` command procedure will not exit until all requested servers have started and initialized. `DMQ$STARTUP` uses `DMQ$WAIT_FOR_SERVICE.EXE`, which accepts one of the following keywords: `COM`, `MRS`, `SBS`, `EVL`, `JRN`, `DECNET`, `TCPIP`, `QTRANSFER`, `CLS`, `NA`, or `ALL`. If the service was specified for startup in `DMQ$INIT.TXT`, then `DMQ$WAIT_FOR_SERVICE` will block until the service has indicated to BEA MessageQ that it is ready, or until it times out (two minutes).

Redirecting Configuration and Log Files

The user area and log area parameters allow you to redirect the location of the group's configuration and log files. Usually, you do this if you want the group to share configuration files with another group or for performance reasons. See Chapter 3, "Configuring Cross-group Connections" for more information on cross-group communications.

If these parameters are not specified, `DMQ$STARTUP` uses the files in the following directories where the variable `bbbb` is the 4-digit bus ID and the variable `ggggg` is the 5-digit group ID.

- `DMQ$DISK:[DMQ$V50.USER.bbbb_ggggg]` for the user area `DMQ$USER`
- `DMQ$DISK:[DMQ$V50.LOG.bbbb_ggggg]` for the log area `DMQ$LOG`
- `DMQ$DISK:[DMQ$V50.MRS.bbbb_ggggg]` for the MRS area `DMQ$MRS`

2 *Defining the Message Queuing Environment*

3 Configuring Cross-group Connections

To enable message queuing between different systems in a network, you must create BEA MessageQ message queuing groups on each system and establish cross-group connections. This chapter discusses cross-group connections.

Note: For detailed information on creating message queuing groups using the Customize utility, refer to Chapter 9, “BEA MessageQ Main Menu and Utilities.”

The following topics are covered in this chapter:

- Connecting to Other Message Queuing Groups
- Configuring the Cross-group Connection Table
- Using Message Routing
- Selecting the TCP/IP Link Driver
- Sharing Group Configuration Files
- Suppressing DECnet Intrusion Alarms
- Configuring DMQ\$GMT_OFFSET for Network Communications

Connecting to Other Message Queuing Groups

Messages can be exchanged between message queuing groups through:

- Cross-group configuration
- Message routing

Groups can exchange messages either directly or indirectly. Table 3-1 describes the message exchange methods.

Table 3-1 Message Exchange Methods

Method	Description	Using
Direct	Between systems that share a network link	Cross-group connection table
Indirect	Between nodes that do not share a network link	Message routing with either BEA MessageQ routing table, <code>DMQ\$INIT.TXT</code> , or dynamic tables

Configuring the Cross-group Connection Table

The following topics are covered in this section:

- Cross-group Connection Table Overview
- Loading the Configuration Data
- Using Cross-group Connection Table Fields

- Table Entry Guidelines

Cross-group Connection Table Overview

The cross-group connection table provides link drivers with the information needed to connect to other BEA MessageQ message queuing groups through DECnet or TCP/IP network communications.

Loading the Configuration Data

You can load this static routing table either by using the startup or by running the `DMQ$LOADER` utility (refer to Chapter 9, “BEA MessageQ Main Menu and Utilities”).

When a queuing group is started, it parses the cross-group connection table and attempts to establish DECnet and TCP/IP links with all the other groups listed in the table. The link drivers maintain these links by reconnecting to them as specified by a timer that is set for each link.

The link drivers also attempt to reconnect to all the groups defined in the cross-group connection table. A request for reconnection can be sent to the link drivers using the `MONITOR` utility (see Chapter 10, “Using BEA MessageQ System Management Utilities”) or link management messages (see the *BEA MessageQ Programmer’s Guide*).

Using Cross-group Connection Table Fields

The cross-group connection table in Listing 3-1 begins with `%XGROUP` and contains the fields shown in Table 3-2. Note that you can enter `-1` in any field to designate the default entry.

3 Configuring Cross-group Connections

Listing 3-1 Cross-group Connection Table

```
-----
%XGROUP      ***** Cross-Group Connection Table *****
*
*
*          gen buf   buf   recon   dly win transport
* gname    gnumber  node    cnt warn pool   sec   sec msg  type  endpt
*-----
*GR1_TCP      1 ipaddr1.company.com  Y  -1 175   -1   -1 -1  TCPIP 2001
*GR1_TCP      1 ipaddr2.company.com  Y  -1 175   -1   -1 -1  TCPIP 2001
*GR1_DECNET   1 daddr1                Y  -1  -1   -1   -1 -1  DECNET
*GR1_DECNET   1 daddr4                Y  -1  -1   -1   -1 -1  DECNET
*
*GR2_TCP      2 ipaddr3.company.com  Y  -1 175   -1   -1 -1  TCPIP 2002
*GR1_TCP      1 ipaddr3.company.com  Y  -1 175   -1   -1 -1  TCPIP 2002
*GR2_DECNET   2 daddr4                Y  -1  -1   -1   -1 -1  DECNET
*GR2_FAILOVER 2 daddr5                Y  -1  -1   -1   -1 -1  DECNET
*
*
%EOS
```

Table 3-2 Cross-group Connection Table Fields

Field	Description
Gname	Specifies the BEA MessageQ internal group name that is used to generate the include files. The BEA MessageQ group name is a 1- to 16-character name.
Gnumber	Specifies the BEA MessageQ internal number of the group. This number must be between 1 and 32000. The group number must be unique on the BEA MessageQ bus. Multiple groups can be defined in this table with the same group number for the purpose of failover; however, only one, unique group number can be active on the BEA MessageQ bus at any one time. See Chapter 14, "Managing Failover," for information on failover configuration.

Configuring the Cross-group Connection Table

Node	<p>Specifies the name of a system to connect to. Successive lines with various system names specify the backup systems for the group. A total of three lines for each transport may be specified. The first entry for each group determines the type of network transport used as the outgoing link.</p> <p>For DECnet software, the system name is a 1 to 6-character node name that is not case sensitive.</p> <p>For TCP/IP software, the system name is a 1 to 64-character node name that is case sensitive.</p>
Gen Cnt	<p>Specifies the generate connection field that indicates how cross-group connections are reestablished after a network failure.</p> <p>N—Never generate an outgoing connection.</p> <p>Y—Automatically attempt to reconnect using the reconnect timer.</p> <p>D—Connections are disabled. No connection can occur until explicitly enabled using the LINK_MGT message. Refer to <i>BEA MessageQ Programmer's Guide</i>.</p>
Buf Warn	<p>Also known as Threshold. Specifies the value for dynamic memory consumption at which congestion control will begin for this cross-group connection. This is not supported in MessageQ for OpenVMS version 4.0A or version 5.0.</p>
Buf Pool	<p>Specifies the optional parameter that overrides the default write buffer pool size for a communications link. It is specified in 1024-byte increments and the default value is 256 (giving a total of 256 x 1024 bytes). This value may need to be changed to accommodate large messages (up to 4MB) as described in Chapter 11, "Sizing and Tuning the BEA MessageQ Environment," under "Buffer Pool Parameter."</p>
Recon	<p>Specifies how often the link drivers attempt to reconnect to a communications link. You can specify separate time values for each network link. This parameter is specified in seconds. The following are valid settings for this parameter:</p> <ul style="list-style-type: none">■ -1 Default—Default time of 30 seconds■ 0 OFF—Only attempts to connect during startup and when explicitly requested■ ≥15 Timer—Time, in seconds, to retry the remote link connection
Dly	<p>Specifies the delay, in seconds, that a sender must wait before using a new window when the receiver is congested. The default value for this field is 1. Its minimum value is 0 and maximum is 2147483647.</p>

3 *Configuring Cross-group Connections*

<code>Win</code>	Specifies the maximum transmission window size, in messages, for this cross-group link. The default value for this field is 250.
<code>Transport Type</code>	Indicates the transport mechanism used for cross-group connections. Valid entries are <code>DECNET</code> and <code>TCPIP</code> .
<code>Endpt</code>	For transport type <code>TCPIP</code> , this field is the TCP/IP socket address (port number) endpoint. This port number ranges from 1024 to 65535 inclusive. Using endpoint numbers under 5000 is recommended. For <code>DECNET</code> transport this field is unused.

Table Entry Guidelines

The local group in the cross-group connection table must have data entered for each transport mechanism used within the group. You can use DECnet or TCP/IP as transport mechanisms. When TCP/IP is defined as the transport for the local group that is booting, the COM Server automatically starts up the TCP/IP link driver. When DECnet is specified, the COM Server automatically starts up the DECnet link drivers.

There are two types of DECnet transports:

- A DECnet transport integral to the COM server which uses the DECNETV2 protocol. The DECNETV2 protocol is only used to communicate with other BEA MessageQ COM Servers on OpenVMS systems.
- A separate DECnet link driver process (`DECNET_LD`) which uses the DECNETV3 protocol. The DECNETV3 protocol is required to communicate with BEA MessageQ servers on UNIX or Windows NT, as well as on OpenVMS systems. The DECNETV3 protocol is the recommended and preferred DECnet transport because it offloads the COM Server from this activity, increasing overall message throughput.

Under normal situations, the BEA MessageQ protocol handshake correctly chooses between these two transports and establishes the link. Specify `DECNET` as the transport section of the Cross-group Connection table to allow BEA MessageQ to determine which DECnet link driver to use.

You can explicitly choose a specific driver for a Cross-group connection by specifying DECNETV2 or DECNETV3 as the transport type rather than DECNET. DECNETV2 selects the COM Server for the transport, while DECNETV3 selects the DECNET link driver (DECNET_LD). DECNETV2 is not supported when attempting to connect to a non-VMS platforms.

There can be a maximum of three entries for the same group/transport pair. Listing 3-2 shows the cross-group connection table for a group named CENTRAL with a group number of ten. There is a maximum of three possible hosts for a group named GROUP1.

Listing 3-2 Maximum Entries in Cross-group Connection Table

```
-----
%XGROUP      ***** Cross-Group Connection Table *****
*
*
*          gen buf   buf   recon   dly win transport
*          cnt warn pool   sec     sec msg type  endpt
*-----
CENTRAL      10     myhost   Y   -1   -1   -1   -1  -1  TCPIP 10010
GROUP1       1     host1    Y   -1   -1   -1   -1  -1  TCPIP 10001
GROUP1       1     host2    Y   -1   -1   -1   -1  -1  TCPIP 10001
GROUP1       1     host3    Y   -1   -1   -1   -1  -1  TCPIP 10001
*
*
%EOS
-----
```

Remote groups that are not listed in the Cross-group Connection Table can connect to your group if the XGROUP_VERIFY parameter in the Profile section is set to NO. In the previous example, if cross-group verify is disabled, connections from GROUP1 on any host will be accepted by CENTRAL. To limit access by unlisted groups, set XGROUP_VERIFY to YES. (Refer to Chapter 13, “BEA MessageQ Security,” for more information on security.)

Using Message Routing

BEA MessageQ uses message routing to exchange messages between systems that do not have a direct network link.

The following topics are covered in this section:

- Routing Tables
- Configuring and Loading the Tables
- DMQ\$INIT Routing Section

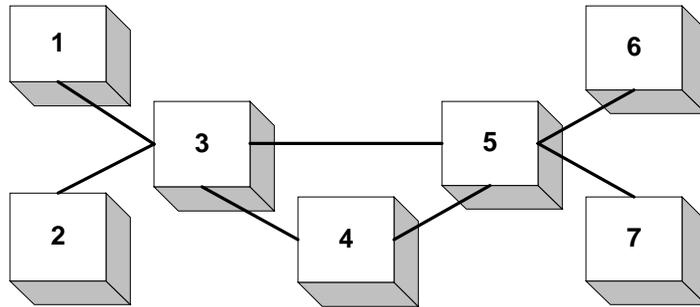
Routing Tables

BEA MessageQ message routing uses either static or dynamic tables as shown in the following table:

Table 3-3 Message Routing Methods

Method	Description
Static tables	These are the cross-group connection table and routing table of the <code>DMQ\$INIT.TXT</code> file.
Dynamic tables	These tables are built out of the incoming cross-group messages from previously unknown groups. Through the process of route discovery, the local system learns the return path of a message sent from an unknown group.

Figure 3-1 shows a sample configuration requiring full message routing capabilities. It describes the next hop that a message must take to reach the target group. It contains three message server groups capable of performing message routing. Each server group has a direct connection to each of the other server groups. The client groups each have a single connection to a server group. They depend on the server group to route all message traffic to the correct target group and learn the backlink from the route messages they are sending back.

Figure 3-1 Sample System Routing Configuration

Groups 3, 4, and 5 are server groups that provide message routing services to the other groups.

Configuring and Loading the Tables

Groups can be defined in either the cross-group connection table or the routing table but not both. Duplicated entries are automatically deleted from the routing table.

All routing information entered in cross-group connection table and the routing table is loaded when the group starts up. There are two ways the routing table is loaded: using either the `DMQ$LOADER` or route discovery.

Once you have determined your routing configuration, you can create the appropriate entries in the cross-group connection and routing table sections of the `DMQ$INIT.TXT` file. Create a routing table for each group by editing the routing table section of the `DMQ$INIT.TXT` file.

All routing information entered in the cross-group connection and routing table sections is loaded when the group starts up.

DMQ\$INIT Routing Section

The %ROUTE section of the DMQ\$INIT.TXT file is used to load the initial routing configuration. This section contains two columns of information as described in Table 3-4.

Table 3-4 %ROUTE Section Column Definitions

Column	Purpose
Target group	Specifies the number of the remote group that is the intended destination of messages.
Route-thru-group	Lists the group number which forms the network link between the local group and the remote target group.

Note: Refer to “Editing DMQ\$INIT to Configure a Group” on how to access and edit the cross-group connection table of the DMQ\$INIT.TXT file.

For example, in order to build a DMQ\$INIT.TXT file for the group 3 (configuration shown in Figure 3-1), entries are needed for all groups directly connected to it. All other groups will require an entry in the Routing Table in order to compute the next hop for the message.

Figure 3-3 shows a sample of the cross-group connection and routing table sections.

Listing 3-3 4-3 Sample Routing Table

```
-----
%XGROUP      ***** Cross-Group Connection Table *****
*
*
*          gen buf   buf   recon   dly win  transport
*gname      gnumber  node   cnt warn pool   sec  sec msg  type  endpt
*-----
Server_1      3   Moe      Y   -1   -1   -1   -1  -1  DECNET
Server_2      4   Larry     Y   -1   -1   -1   -1  -1  DECNET
Server_3      5   Curly     Y   -1   -1   -1   -1  -1  DECNET
*
*Client_1     1   Sleepy    N   -1   -1   -1   -1  -1  DECNET
*Client_2     2   Doc       N   -1   -1   -1   -1  -1  DECNET
*Client_3     6   Dopey     N   -1   -1   -1   -1  -1  DECNET
*Client_4     7   Grumpy    N   -1   -1   -1   -1  -1  DECNET
```

```
*
%EOS
*
%ROUTE * initial routing table
*
* Target      Route-Thru
* Group       Group
* -----
   1          3 * Group 1 is routed thru Group 3
   2          3 * Group 2 is routed thru Group 3
   6          5 * Group 6 is routed thru Group 5
   7          5 * group 7 is routed thru Group 5
%EOS
```

Selecting the TCP/IP Link Driver

BEA MessageQ for OpenVMS offers link drivers for several popular TCP/IP products including:

- Compaq TCP/IP Services for OpenVMS (formerly called UCX)
- Process Software Corporation (PSC) TCPware
- Process Software Corporation (PSC) MultiNet

TCP/IP link drivers are mutually exclusive; therefore, you must select one driver. The `DMQ$TCP_IP_LD` logical name indicates the selection and must exist prior to starting the group. This logical can be defined manually, or is defined automatically by the `DMQ$BOOT.COM` procedure. Define the `DMQ$TCP_IP_LD` logical name as `DEC` to start the DEC/UCX link driver, `PSC` to start the TCPware link driver, or `TGV` to start the MultiNet link driver.

You can edit the `DMQ$BOOT.COM` procedure to start any of these drivers by changing the value associated with the `DMQ$TCP_IP_LD` logical name. Access the `DMQ$BOOT.COM` procedure using the `CUSTOMIZE` option on the BEA MessageQ main menu. Refer to Chapter 9, “BEA MessageQ Main Menu and Utilities,” or the `DMQ$STARTUP` procedure in Chapter 2, “Defining the Message Queuing Environment.”

Edit the `DMQ$BOOT.COM` file, changing the commenting in the following lines:

3 *Configuring Cross-group Connections*

```
$ tcpip_ld      = "DEC"      ! Use the DEC/UCX TCP/IP LinkDriver
$!*** tcpip_ld = "TGV"      ! Use the Multinet TCP/IP LinkDriver
$!*** tcpip_ld = "PSC"      ! Use the TCPware TCP/IP LinkDriver
$!*** tcpip_ld = " "        ! Select TCP/IP LD at SYSTEM level
```

The default setting for DMQ\$TCPIP_LD is DEC. Change the commenting in the file to change the setting. Use a pair of quotation marks with no value (" ") to suppress the creation of the logical name in the BEA MessageQ logical name table. This allows the logical to be created at the system level, thereby, controlling multiple groups simultaneously.

For example, you select link drivers at the system level as follows:

```
$ DEFINE/SYSTEM/EXEC DMQ$TCPIP_LD "DEC"      ! DEC TCP/IP

or

$ DEFINE/SYSTEM/EXEC DMQ$TCPIP_LD "PSC"      ! TCPWARE
```

Sharing Group Configuration Files

Groups can share the same configuration, log, and MRS directories. Sharing configuration files is convenient when many groups have the same configuration. This is common in VMScluster environments, where there are groups running on each cluster member.

Note that the names of the log and MRS files of each group contain the bus ID and group ID of each message queuing group. These tags in the file names allow you to distinguish the files even if they are in the same directory as another group's files.

Listing 3-4 shows how to start a group with group ID 5, bus ID 4 with the same configuration, log and script files as a group with group ID 6, bus ID 4.

Listing 3-4 Sharing Configuration Files

```
$ @DUAL:[DMQ$V50.EXE]DMQ$SET_LNM_TABLE 4 6
%DMQ-S-SETLNM, Set to MessageQ
LNM table DMQ$LNM_0004_00006
$ sho logical DMQ$USER
  "DMQ$USER" = "DMQ$DISK:[DMQ$V50.USER.0004_00006]"
```

```
(DMQ$LNM_0004_00006)
$ sho logical DMQ$LOG
  "DMQ$LOG" = "DMQ$DISK:[DMQ$V50.LOG.0004_00006]"
(DMQ$LNM_0004_00006)
$ @DMQ$EXE:DMQ$STARTUP 4 5 " Y " -
DMQ$DISK:[DMQ$V50.USER.0004_00006] -
DMQ$DISK:[DMQ$V50.LOG.0004_00006]
```

When message queuing groups share the same `DMQ$USER` area, they execute the same `DMQ$BOOT.COM` command procedure that defines all the logical names. If you make no changes to `DMQ$BOOT.COM` file, the logical name `DMQ$MRS` is the same for those groups, and the MRS files of the groups will be on the same disk.

Note: Locating MRS files of multiple groups on a common disk can have significant performance impact. Refer to Chapter 5, “Configuring Message Recovery,” for more information.

Suppressing DECnet Intrusion Alarms

If you attempt a cross-group connection to a nonactive group using the DECnet transport, BEA MessageQ attempts to start a default DECnet object. If the DECnet default account has been eliminated from the system on which the group is running, DECnet intrusion alarms are generated every time a BEA MessageQ group attempts to connect to a nonactive group.

You can eliminate DECnet intrusion alarms by creating a dummy DECnet object to handle the connection attempt. The dummy DECnet object is configured to accept and then immediately drop the connection.

The following steps describe how to create a dummy DECnet object:

1. Use the OpenVMS `AUTHORIZE` utility to create an account for the dummy DECnet object. This account should be set up for network-only access, with `TMPMBX` and `NETMBX` privileges.

When BEA MessageQ binds to the named object, it is going to use its own security information (for example, that of the current process). Use the user, password, and file name of this process when the COM Server is not available.

3 Configuring Cross-group Connections

2. Add an entry to the NCP database.

The BEA MessageQ DECNETV3 protocol object is of the form `DMQ_bbbbggggg` and the DECNETV2 protocol object is in the form of `DMQbbbbggggg`, where `bbbb` is the bus number, zero-filled to the left, and `ggggg` is the group number, also zero-filled to the left. The DECNETV3 protocol object identifies the bus and group number in hexadecimal format. The DECNETV2 protocol object is represented in decimal format.

For example, for bus 50 group 35, the DECNETV2 protocol object name is `DMQ005000035`.

3. Load the dummy object into the permanent database using the following NCP command:

```
NCP> DEFINE OBJECT DMQ005000035 NUMBER 0 -  
_ USER      user_id -  
_ PASSWORD password FILE REJECT.COM
```

4. While BEA MessageQ is *not* running, load the object into the volatile database using the following NCP command:

```
NCP> SET OBJECT DMQ005000035 ALL
```

5. Restart BEA MessageQ so that it can bind itself to the predefined DECnet object. When the target group is not running, inbound DECnet connection requests are handled in a valid account by the `REJECT.COM` command procedure. You must create `REJECT.COM` and supply the following content:

```
$ OPEN X SYS$NET  
$ CLOSE X  
$ EXIT
```

When this command procedure executes, the inbound connection is established and the link is dropped immediately. This procedure allows the remote system to know you are not accepting new BEA MessageQ connects immediately, but rather you are waiting for the outgoing timer on the remote end to expire. You can change the contents of the `REJECT.COM` command procedure to contain only the `$ EXIT` statement, however, connection timeouts will take longer.

Configuring DMQ\$GMT_OFFSET for Network Communications

If you are using networking software, set the value of the BEA MessageQ logical name DMQ\$GMT_OFFSET. DMQ\$GMT_OFFSET is defined as the difference in hours between the time zone of the local system and Greenwich mean time (GMT). This value is used by the cross-group protocol and can seriously affect throughput if set improperly. Table 3-5 lists the GMT offsets for most time zones around the world.

Table 3-5 GMT Offsets

Geographic Area	Offset	Daylight Offset
Western Europe	0	- 1
Central Europe	- 1	- 2
Eastern Europe	- 2	- 3
Eastern U.S.	+5	+4
Central U.S.	+6	+5
Mountain U.S.	+7	+6
Pacific U.S.	+8	+7
Alaska U.S.	+9	+8
Hawaii U.S.	+10	+9
Japan	- 9	.
Hong Kong	- 8	.
Singapore	- 8	.

For example, to configure a system set to the Eastern Standard Time (EST), enter the following command:

```
$ DEFINE/SYSTEM DMQ$GMT_OFFSET "+5"
```

3 *Configuring Cross-group Connections*

4 Configuring Message Queues and Global Memory

BEA MessageQ stores messages in global sections and stores pointers to the messages in queues. Developers are primarily concerned with queue configuration. BEA MessageQ system managers configure the global sections where the messages are stored.

This chapter describes how to configure message queues and global sections. See the *BEA MessageQ Introduction to Message Queuing* for a description of message queues.

The following topics are covered in this chapter:

- Configuring Message Queues
- Configuring Global Memory

Configuring Message Queues

You can configure your message queues using the queue configuration table section of the `DMQ$INIT.TXT` file. Listing 4-1 is a sample of the queue configuration table section. You can access this table using the `CUSTOMIZE` option on the BEA MessageQ main menu. For more information on the main menu options, refer to Chapter 9, “BEA MessageQ Main Menu and Utilities.”

4 Configuring Message Queues and Global Memory

Use this table to configure your message queues following the configuration guidelines discussed in the section “Queue Configuration Table Rules.”

Listing 4-1 Queue Configuration Table Section

```
%QCT          ***** Queue Configuration Table *****
*
*
*      ---Pool Quota---  UCB  Q  Q  Confirm Perm Name  Chk
*      Queue Name      Num  Bytes Msgs Ctrl Send Type Own  Style Act  Scope ACL
*-----
*      Sample Queues
QUEUE_1      1    64000  100 All    .    .    .    II    Y    L    N
QUEUE_2      2    64000  100 Msg    .    .    .    EI    Y    L    N
QUEUE_3      3    64000  100 Byte  .    .    .    EO    Y    L    N
QUEUE_4      4    64000  100 None  .    .    .    .    Y    L    N
QUEUE_5      5    64000  100 .    .    .    .    Y    L    N
QUEUE_6      6    64000  100 .    .    .    .    Y    L    N
QUEUE_7      7    64000  100 .    .    .    .    Y    L    N
QUEUE_8      8    64000  100 .    .    .    .    Y    L    N
QUEUE_9      9    64000  100 .    .    .    .    Y    L    N
QUEUE_10     10   64000  100 .    .    .    .    Y    L    N
*
*      SBS Server uses the following UCB numbers for Optimized Delivery
*
SBS_ETH_CONTROL      74      0      0    .    E    .    .    .    .    L    N
SBS_ETH_CHAN1        75      0      0    .    E    .    .    .    .    L    N
SBS_ETH_CHAN2        76      0      0    .    E    .    .    .    .    L    N
*
*      Queues 90-100 & 150-199 are reserved for MessageQ utilities
TEMPORARY_Q          0    64000  100 .    .    .    .    .    L    N
SPARE1               90   100000  100 .    .    .    .    Y    L    N
ALL_UCBS             91      0      0    .    .    .    .    .    L    N
TIMER_QUEUE          92      0      0    .    .    .    .    .    L    N
NULL                 93      0      0    .    .    .    .    .    L    N
NA_SERVER            94  1000000  1000 None .    .    .    .    N    L    N
QTRANSFER_SERVER    95  1000000  1000 None .    .    .    .    N    L    N
DEAD_LETTER_QUEUE   96    64000  100 .    .    .    .    Y    L    N
MRS_SERVER           98  1000000  1000 None .    .    .    .    N    L    N
SBS_SERVER           99  1000000  1000 None .    .    .    .    N    L    N
COM_SERVER          100  1000000  1000 None .    .    .    .    N    L    N
DCL_BY_Q_NAME       151      0      0    .    .    .    .    .    L    N
TCPIP_LD            152  1000000  1000 None .    .    .    .    N    L    N
DECNET_LD           153  1000000  1000 None .    .    .    .    N    L    N
RESERVED_LD         154  1000000  1000 None .    .    .    .    N    L    N
JRN_SERVER          156  1000000  1000 None .    .    .    .    N    L    N
DMQ_FULLLTEST_PQ   191   250000  100 .    .    .    .    N    L    N
```

```

DMQ_FULLTEST_PQ      192 250000 100 . . S 191 . N L N
EXAMPLE_Q_1          193  64000 100 . . . . . N L N
EXAMPLE_Q_2          194  64000 100 . . . . . N L N
IVP_unowned_sq       195 250000 100 . . S . . N L N
*
%EOS

```

Queue Configuration Table Parameters

The following table explains the parameters (fields) in the queue configuration table.

Table 4-1 Queue Configuration Table Fields

Field Name	Description
Queue Name	<p>The BEA MessageQ internal name of the message queue. The queue name is 1 to 255 characters long and is stored as uppercase. The name is stored in the group's name table available for run-time translation by the queue location services.</p> <p>If the CUSTOMIZE procedure is run and you request that language specific files be created, the queue names are added to <code>p_proc.h</code> and are prefixed with <code>PAMS_</code>.</p>
Num	<p>The BEA MessageQ internal queue number that ranges from 0 to 3949 and 4000 to 6000, inclusive. The queue number must be below the value defined by <code>FIRST_TEMP_QUEUE</code> in the <code>%PROFILE</code> section. Multipoint outbound target (MOT) queue values are not allowed in the Queue Configuration Table.</p>

4 Configuring Message Queues and Global Memory

Table 4-1 Queue Configuration Table Fields

Pool Quota	<p>Contains three parameters: <code>Bytes</code>, <code>Msgs</code>, and <code>Ctrl</code></p> <ul style="list-style-type: none">■ <code>Bytes</code>—The maximum number of data bytes sent to a message queue. The range is from 0 to $2^{31}-1$ (2147483647). A period (.) specifies the default which is 64000.■ <code>Msgs</code>—The maximum number of messages that can be pending to a queue. The range is from 0 to $2^{31}-1$ (2147483647). A period (.) specifies the default which is 100.■ <code>Ctrl</code>—Controls the enabling and disabling of queue quotas. Valid settings are (case insensitive):<ul style="list-style-type: none">A or All—All queue quotas enabledN or None— All queue quotas disabledB or Byte—Only Byte quota enabled; Msgs quota disabledM or Msgs—Only Msgs quota enabled; Byte quota disabled <p>" . "— A period (.) specifies the default which is All</p> <p>The quotas for the queue may need to be changed to accomodate large messages (up to 4MB) as described in Chapter 11, “Sizing and Tuning the BEA MessageQ Environment” under “Queue Quota.”</p>
UCB Send	<p>Specifies special message processing by using user-writable subroutines. Send user callback points (send UCB) are designated by the uppercase letters A through J. A period (.) means that there are no send UCBs for this queue.</p>
Q Type	<p>Designates the queue type. It can be a primary (P), secondary (S), or multi-reader (M). A period (.) specifies the default which is primary (P). Configure any queue with a send UCB as a primary queue.</p>
Q Own	<p>Identifies queue owner. Used for preallocated, permanent secondary queues, and defines the primary queue with which this queue is to be associated.</p>
Confirm Style	<p>Controls the method by which messages can be confirmed. Valid settings are (case insensitive):</p> <ul style="list-style-type: none">■ II—Implicit, in-order confirmations■ EI—Explicit, in-order confirmations■ EO—Explicit, out-of-order confirmations■ " . "—A period (.) specifies the default, which is EO. <p>For more information on this parameter, refer to Chapter 5, “Configuring Message Recovery.”</p>

Table 4-1 Queue Configuration Table Fields

Perm Act	Indicates whether a permanent queue can accept messages when no process has currently attached to the queue. The queue is activated when a process attaches to BEA MessageQ . This applies to only primary and secondary queues. Multi-reader queues are always active, and temporary queues are only active when a process has attached to the temporary queue. Valid settings are: <ul style="list-style-type: none"> ■ Y for always active ■ N for active on attach ■ period (.) for the default setting (Y for always active)
Name Scope	Indicates the scope in which the DMQ\$LOADER utility advertises the association between names and addresses. Name scope must contain the letter L or G. L indicates that the name is available for lookup only within the message queuing group. G indicates that the name is available to the entire message queuing bus. When the DMQ\$LOADER utility encounters a name scope of G, it advertises the address setting through the BEA MessageQ Naming Agent.
Chk ACL	Indicates whether the queue has controlled read access. When Chk ACL is enabled, any attempt to read from the queue results in an access check against a corresponding file that has limited read access. Valid settings are Y for enable and N for disable. See Chapter 13, “BEA MessageQ Security,” for more information.

Queue Configuration Table Rules

When assembling the queue configuration table, follow these rules:

- Specify queues by queue name and queue number.
- Permanent queue numbers range from 1 to 3949. The upper limit is one less than the FIRST_TEMP_QUEUE value field in the profile section in DMQ\$INIT.TXT.
- Temporary queues can range from 200 to 3999. The minimum temporary queue is controlled by the FIRST_TEMP_QUEUE field in the profile section in DMQ\$INIT.TXT, and the value can range from 200 to 3949. The default is 200.
- Adhere to the following queue naming rules:
 - Queues use all alphanumeric characters plus hyphens (-), underscores (_), and dollar signs (\$). Queue names with dollar signs are only allowed on OpenVMS systems.
 - Queue names are limited to a length of 256 characters.

4 Configuring Message Queues and Global Memory

- Queue names are case sensitive; therefore, `Queue_1` and `QUEUE_1` are considered two different queues.
- Use the Pool Quota Bytes column to specify the number of message bytes pending to a particular queue before the queue requests for that target are rejected.
- Use the Pool Quota Msgs column to specify the number of messages that can be pending to a queue before the queue requests for that target are rejected.
- Use the Pool Quota Ctrl column to specify the enabling/disabling of queue quotas.
- You can specify queue entries in any order. Multiple queue entries can share the same queue number with only the first entry taking effect. These multiple definitions of queues are useful for setting up aliases in the include files and in the group name table (see Chapter 7, “Creating Global Names”).
- Secondary queues do not have to be assigned an owner in `DMQ$INIT.TXT`. Instead, any user that passes access control checks can be attached for read access to the queue. However, only one user is allowed to be attached to a secondary queue at any particular time.
- This feature allows read access to a recoverable queue to be passed from process to process, without requiring those processes to exit from the bus and reattach as a particular primary queue.
- If a secondary queue is assigned an owner in `DMQ$INIT.TXT`, then it is attached when the owner primary queue is attached, if it passes access control checks.
- Specify all temporary queues with a single table entry with the queue name `TEMPORARY_Q`. BEA MessageQ assigns a queue number to a temporary queue when a process requests one. To define the default receive quota and access control for temporary queues, BEA MessageQ equates the queue name `TEMPORARY_Q` to queue number 0.
- BEA MessageQ recoverable messages are removed from the recovery system after they are confirmed by BEA MessageQ or by the receiver application using the `pams_confirm_msg` service. BEA MessageQ offer two types of message confirmation; explicit and implicit. It also allows in-order and out-of-order message confirmation. Confirmation characteristics of the queue are set using the `CONFIRM_STYLE` queue attribute in the `QCT` section. Refer to the *BEA MessageQ Programmer's Guide* for a complete explanation of message confirmation.

- The queue configuration table also accepts SBS broadcast stream entries (4000 to 6000). These entries do not correspond to queues but are useful for creating symbols in include files and the group name table. Refer to the *BEA MessageQ Programmer's Guide* for more information.
- If you use dynamic naming, add alias entries to the group name table (GNT) in the `DMQ$INIT.TXT` file. This allows for easier division of features into separate servers at a later time.

Configuring Global Memory

The BEA MessageQ global memory provides storage for queue names and addresses and allocates memory space for message queuing groups and message buffers.

The following topics are covered in this section:

- How BEA MessageQ Uses Global Memory
- Using the Buffer Pool Configuration Table

How BEA MessageQ Uses Global Memory

BEA MessageQ creates seven global sections for each message queuing group that is running on a node. The four sections for message control are listed in Table 4-2 and the three sections describing the buffer pool configuration are listed in Table 4-3. In the following tables, *bbbb* corresponds to the bus ID and *ggggg* corresponds to the group ID of the message queuing group.

Table 4-2 Message Control Section

Section	Use it to store ...
<code>DMQ\$MCS_C_bbbb_ggggg</code>	General control information
<code>DMQ\$GRP_C_bbbb_ggggg</code>	Cross-group link control information
<code>DMQ\$MRQ_C_bbbb_ggggg</code>	Multireader queue (MRQ) control information

Table 4-2 Message Control Section

<code>DMQ\$GNT_C_bbbb_ggggg</code>	Group Name Table (GNT) information
------------------------------------	------------------------------------

Messages are stored in one of the buffer pool global sections listed in Table 4-3.

Table 4-3 Buffer Pool Global Sections

Global Section	Stored Message Size
<code>DMQ\$LLS_S_bbbb_ggggg</code>	Small
<code>DMQ\$LLS_M_bbbb_ggggg</code>	Medium
<code>DMQ\$LLS_L_bbbb_ggggg</code>	Large

Using the Buffer Pool Configuration Table

To change BEA MessageQ memory allocation, configure the buffer pool in the buffer pool configuration table of the `DMQ$INIT.TXT` file (see Chapter 2, “Defining the Message Queuing Environment,” and Chapter 9, “BEA MessageQ Main Menu and Utilities,” for more information on working with the `DMQ$INIT.TXT` file).

This table controls the size of the buffer pool where messages are stored. A buffer pool consists of fixed-size memory buffers that hold one message each. The buffer pool is created during COM Server startup.

The buffer pool contains space for all messages on a system. It must be sized to contain all messages active in memory at any given time. When this pool is fully allocated, attempts to send a message cause `PAMS__REMQUEFAIL` to be returned to the sending application. This return code can also indicate that insufficient global memory buffers were found to contain a particular message request. If `GROUP_MAX_MESSAGE_SIZE` is larger than a `LARGE` buffer, then multiple buffers are chained together to contain such a message. The maximum `LARGE` buffer size is 65K. However, the default `LARGE` buffer size is 64K.

Listing 4-2 shows the buffer pool configuration table section of `DMQ$INIT.TXT`.

Listing 4-2 Buffer Pool Configuration Table Section

```

-----
%BUFFER      ***** Buffer Pool Configuration Table *****
*
*Msg-Block-Type  Byte-Size      Number      Warning-level      Reserve
*-----*-----*-----*-----*-----
SMALL           256                50           10                 2
MEDIUM          5000               10           2                  1
LARGE           64000              3            1                  0
%EOS
*
-----

```

Table 4-4 describes the fields (parameters) in the buffer pool configuration table.

Table 4-4 Buffer Pool Configuration Table Fields

Field Name	Description
Msg-Block-Type	Indicates whether the buffer pool being specified is for small, medium, or large messages.
Byte-Size	Indicates the size of the largest message that can be stored in the specified buffer pool. The minimum byte size required to boot is 1500. The minimum byte size to support all BEA MessageQ services is 8192. The maximum value is 64000.
Number	Indicates the maximum number of messages desired of this size. The maximum number is 50000.
Warning-level	Indicates the threshold number of buffers remaining before a warning message is sent to the console. The minimum number is zero, and the maximum number must be less than the number of messages.
Reserve Count	Allocates the last <i>n</i> messages in the buffer pool for BEA MessageQ internal use. This allocation allows BEA MessageQ internal messaging to occur when the buffer pool is full. The reserve count space, set at 5% to 10% of the buffer pool capacity, would be a good starting point when allocating the space. By reserving messages, the unblocking messages used by PDEL_MODE_WF_XXX and PAMS_CONFIRM_MSG messages have a higher tolerance to the buffer pool capacity overflow.

4 *Configuring Message Queues and Global Memory*

5 Configuring Message Recovery

BEA MessageQ Message Recovery Services (MRS) provide a mechanism for guaranteed message delivery by storing messages on disk and automatically attempting redelivery until the message is received by the target system. On BEA MessageQ for OpenVMS systems, message recovery is provided by two servers:

- The MRS Server
- The Journal (JRN) Server

The following topics are covered in this chapter:

- How Message Recovery Services Work
- Starting MRS and JRN Servers
- Configuring MRS and JRN Servers
- Sizing MRS File Space
- MRS Internal Operation Tracing at Startup
- Confirming Message Removal from the Recovery System

How Message Recovery Services Work

In BEA MessageQ, the COM Server process starts up subsidiary servers, depending upon the contents of the Profile section (see “Setting Servers and Configuring Groups in the Profile Section”). The server which handles recoverable messaging is the MRS Server. It is started by setting `ENABLE_MRS` to `YES`. The server which handles auxiliary journal files is the Journal (JRN) Server. It is started by setting `ENABLE_JRN` to `YES`.

BEA MessageQ message recovery services (MRS) enable an application to send a message to a target application and guarantee its delivery despite an application, system, or network failure. A message is recoverable if BEA MessageQ writes it to a message recovery journal prior to delivery. BEA MessageQ has two message recovery journals that are controlled by the MRS Server:

- Store and Forward file (SAF)—for outgoing recoverable messages
- Destination Queue file (DQF)—for incoming recoverable messages

In addition, the BEA MessageQ message recovery system provides the following auxiliary journal files (controlled by the JRN Server):

- Dead Letter Journal (DLJ)—a disk-based mechanism for storing undeliverable messages for recovery under user or program control
- Post Confirmation Journal (PCJ)—a file that contains successfully delivered recoverable messages

Using Recoverable Journal Files

If a recoverable message cannot be delivered to the target application, the MRS Server writes it to the SAF or DQF file, depending upon the delivery interest point selected when the message was sent. DQF and SAF files are dynamically created and deleted as needed by the recovery system (`MRS_SERVER` process).

After a message is written to the SAF or DQF file, the MRS Server writes a copy of the message into the message queue, where the receiver program can access it. After the receiver program has completed processing the message, it either reads the next

recoverable message, thereby performing an implicit confirmation (if the queue is configured for this option), or calls the `pams_confirm_msg` service to explicitly instruct MRS to delete the message recovery file.

The size of SAF and DQF files is fixed and is set by the MRS parameter `AREA_SIZE`. The MRS Server creates a new SAF or DQF file each time the file fills up. MRS attempts to empty SAF files by periodically attempting to reach the corresponding remote queue. The MRS Server creates a separate set of SAF files for each remote queue. The maximum number of SAF files that can be created is set using the `NUM_SAF_AREAS` parameter. The maximum number of DQF files that can be created is set using the `NUM_DQF_AREAS` parameter.

In support of large messages, up to 4 MB, the journal files may be automatically extended by the MRS and JRN Servers in order to store a message that will not fit into the "fixed" size. This prevents a large message from being split across multiple journal files.

Both DQF and SAF files can be closed and the stream of messages in the file redirected to other targets. This can be in response to a message to the MRS Server from a suitably privileged user program, or activated from the Utilities menu (`DMQ$MGR_UTILITY`) as described in Chapter 10, "Using BEA MessageQ System Management Utilities."

Using Auxiliary Journal Files

The DLJ and PCJ journals are served by a separate server process called `JRN_SERVER` that maintains a history of all journal filename changes in a disk file. This history is displayed by the `DIR` and `DUMP` commands of `DMQ$MGR_UTILITY` to make it easy to find an old journal.

The PCJ and DLJ journal filename formats are defined in the `%MRS` section of `DMQ$INIT.TXT` as follows.

```
PCJ_FILENAME  DMQ$MRS:MRS_%bg.PCJ ! char[64] - %bg is a macro that
DLJ_FILENAME  DMQ$MRS:MRS_%bg.DLJ ! char[64] - expands to bus_group
```

BEA MessageQ adds a sequence and version number to the file names for file management purposes. The resulting default PCJ and DLJ file names appear as follows on disk:

```
DMQ$MRS:MRS_bbbb_ggggg_xxxxxx.PCJ;ver
DMQ$MRS:MRS_bbbb_ggggg_xxxxxx.DLJ;ver
```

5 Configuring Message Recovery

In these filenames, *bbbb* is the bus number, *ggggg* is the group number, *xxxxxx* is a sequence number assigned by the JRN server, and *ver* is an OpenVMS file version number assigned by the JRN server. The journals use the sequence number to tie multiple journal files from the same stream together. The version number is used to distinguish between different journal streams.

Note: The file extension is user-defined. By default, the `.PCJ` and `.DLJ` extensions are used.

The application does not have to account for the internal sequence number (*xxxxxx*) added to the file names when using the `pams_open_jrn` API call.

Listing 5-1 shows a sample journal history. Note that the internal sequence number is not displayed in the listing.

Listing 5-1 Example Journal History

```
$type DMQ$MRS:DMQ_0099_00040_DLJ.JH
  1-DEC-1997 12:58:45.70 DMQ$MRS:MRS_0099_00040_*.DLJ;1
  6-JAN-1998 10:41:01.27 DMQ$MRS:MRS_0099_00040_*.DLJ;2

$type DMQ$MRS:DMQ_0099_00040_PCJ.JH;1
  1-DEC-1997 12:58:47.80 DMQ$MRS:MRS_0099_00040_*.PCJ;1
  6-JAN-1998 10:42:06.40 DMQ$MRS:MRS_0099_00040_*.PCJ;2
  6-JAN-1998 10:42:12.97 DMQ$MRS:MRS_0099_00040_*.PCJ;3
```

Using the Dead Letter Journal

A Dead Letter Journal (DLJ) can be configured for each BEA MessageQ message queuing group. On BEA MessageQ for OpenVMS systems, DLJ files are created automatically by the JRN Server when a message queuing group is configured with JRN enabled.

To use the DLJ, the sender program uses the `pams_put_msg` service specifying the appropriate delivery mode argument and `PDEL_UMA_DLJ` as the `uma` argument. Messages that are forced to take the UMA because they cannot be delivered are written to the DLJ of the sender's group. Both recoverable and nonrecoverable messages can specify the DLJ UMA.

The DLJ file provides a backup mechanism for recovering messages that could not be delivered to their target queue. On OpenVMS systems, an application can provide recovery under program control by opening the DLJ, reading each message in the file and attempting redelivery by using the `pams_open_jrn`, `pams_read_jrn`, and `pams_close_jrn` services.

Using the Post Confirmation Journal

Recoverable messages that are processed by the receiver program can be written to the post confirmation journal (PCJ) of the target group. On OpenVMS systems, the receiver program uses the `force_j` argument of the `pams_confirm_msg` service to control whether successfully delivered messages are journaled as follows:

- `PDEL_DEFAULT_JRN`—enables writing the message to the PCJ file if the PCJ is set enabled
- `PDEL_FORCE_JRN`—enables writing to a journal file regardless of whether the group was configured with journaling enabled
- `PDEL_NO_JRN`—disables journaling regardless of the PCJ settings

The default journaling action can be set under program control by the `MRS_SET_PCJ` message to the MRS Server process. The current PCJ file can also be closed and a new one opened by the same message.

Retrieving Journaled Messages

To retrieve journaled messages on BEA MessageQ for OpenVMS systems, use the `pams_open_jrn`, `pams_read_jrn`, and `pams_close_jrn` services to open, read, and close the DLJ or PCJ files. Each journal file is assigned a journal handle by the `pams_open_jrn` routine. This handle is passed to `pams_read_jrn` and `pams_close_jrn` to identify the journal file.

Each successive journal read operation returns the next message. Arguments for `pams_read_jrn` are similar to arguments for `pams_get_msg` with the following additional message data that identifies the target of the message:

- Message queuing group and queue address of the target
- Confirmation value used by target application (PCJ only)
- Date and Time message was written to file.

After a message is retrieved from a journal file, it can be reprocessed, or reinserted into the message flow using the `pams_put_msg` service.

Starting MRS and JRN Servers

To use the message recovery services, set the following parameters in the profile section of the `DMQ$INIT.TXT` file as shown in Listing 5-2:

- Set the symbol `ENABLE_MRS` to `YES` to start the MRS Server.
- Set the symbol `ENABLE_JRN` to `YES` to start the JRN Server.

Listing 5-2 is an excerpt from the profile section (see Chapter 2, “Defining the Message Queuing Environment,” for more information on BEA MessageQ startup).

Listing 5-2 DMQ\$INIT.TXT Profile Section Enable Parameters

```
-----  
%PROFILE          ***** Profile Parameters *****  
*  
...  
ENABLE_MRS YES ! Enable MessageQ Message Recovery Services  
ENABLE_JRN YES ! Enable Dead letter / Post Confirmation journal  
...  
*  
%EOS  
-----  
-----
```

Configuring MRS and JRN Servers

The following topics are covered in this section:

- Setting Parameters for MRS and JRN

- How BEA MessageQ Manages Destination Queue Files (DQFs)
- Specifying the Location of MRS Files

Setting Parameters for MRS and JRN

You can control the operation of MRS and JRN Servers by setting the parameters in the MRS/JRN initialization section of `DMQ$INIT.TXT`. Listing 5-3 shows typical MRS and JRN initialization entries in the `DMQ$INIT.TXT` file.

Listing 5-3 MRS and JRN Server Initialization Section

```
%MRS      ***** MRS/JRN Servers Initialization Section *****
*
AREA_SIZE          512  ! disk blks per file (min:128, max:16384, def:512)
NUM_DQF_AREAS     1000 ! min:100, max:1000000,      def:1000
NUM_SAF_AREAS     1000 ! min:  0, max:1000000,      def:1000
NUM_PCJ_AREAS     1000 ! min:  0, max:1000000,      def:1000
NUM_DLJ_AREAS     1000 ! min:  0, max:1000000,      def:1000
NUM_MESSAGES      512  ! min:128, max:2147483647,  def:512
NUM_QUEUES        128  ! min:128, max:2147483647,  def:128
CACHE_PERCENTAGE  90   ! % rcv quota for MRS msgs (min:1, max:100, def:90)
*
USE_HIGH_WATER_MARK  YES ! checkpt MRS sizing params to disk      (YES/NO)
LOAD_MRS_CTRS       YES ! init recoverable msg ctrs on startup  (YES/NO)
RCVR_ONLY_CONFIRM   YES ! limit confirms to receiving process   (YES/NO)
*
XGRP_JRN_CTRL       NO  ! allow JRN cntrl msgs from other groups (YES/NO)
REDELIVERY_TIMER    10  ! integer seconds (min:0, max:5000, def:10)
*
PCJ_FILENAME        DMQ$MRS:MRS_%bg.PCJ ! char[64] - %bg is a macro that
DLJ_FILENAME        DMQ$MRS:MRS_%bg.DLJ ! char[64] - expands to bus_group
*
%EOS
```

Table 5-1 describes the MRS and JRN Server initialization parameters (fields). Some parameters may need to be changed to accommodate large messages (up to 4MB). See “Configuring BEA MessageQ for Large Messages” in Chapter 11, “Sizing and Tuning the BEA MessageQ Environment,” for more information.

Table 5-1 MRS/JRN Initialization Fields

Field	Description
AREA_SIZE	Number of disk blocks in a recovery file area. Minimum = 128, maximum = 16,384, default = 512
NUM_DQF_AREAS	The number of areas that can hold local recoverable messages. Minimum = 100, maximum = 1000000, default = 1000
NUM_SAF_AREAS	The number of areas that can hold remote recoverable messages. Minimum = 0, maximum = 1000000, default = 1000
NUM_PCJ_AREAS	The number of areas that can hold Post Confirmation Journal files. Minimum = 0, maximum = 1000000, default = 1000
NUM_DLJ_AREAS	The number of areas that can hold Dead Letter Journal files. Minimum = 0, maximum = 1000000, default = 1000
NUM_MESSAGES	Sets the number of messages (both remote and local) that MRS can retain in memory. Carefully review this setting and set the value of this parameter to the most realistic number possible. When this parameter is exhausted, the MRS Server will stop and allocate more memory and then continue processing. As the MRS Server grows, performance degrades. Minimum = 128; maximum = 2147483647; default = 512
NUM_QUEUES	Sets the number of queues that MRS can track. Minimum = 128; maximum = 2147483647; default = 128

CACHE_PERCENTAGE	<p>The receive quota that determines the number of messages MRS maintains queued in the message pool to a given target queue. Minimum = 1, maximum = 100, default = 90.</p> <p>Note: We recommend that you always set CACHE_PERCENTAGE to less than 100.</p> <p>This parameter operates as follows:</p> <ul style="list-style-type: none">■ When a message queue is attached, the number of recoverable messages delivered either matches or exceeds this limit.■ When the limit is reached, no more recoverable messages are delivered until enough recoverable messages are confirmed by the target to drop below the limit.■ While recoverable message delivery is stopped due to the limit, new recoverable messages are written to the DQF file to be delivered in order of receipt. Nonrecoverable messages are delivered as usual.
USE_HIGH_WATER_MARK	<p>The parameter that controls automatic MRS sizing on the disk (YES/NO). Set USE_HIGH_WATER_MARK = YES to size MRS from the largest values saved from previous runs. Set USE_HIGH_WATER_MARK = NO if sizing MRS from largest previous size causes problems at startup and causes MRS to purge its checkpoint database.</p>
LOAD_MRS_CTRS	<p>The parameter that initiates recoverable message counting prior to startup (YES/NO).</p>
RCVR_ONLY_CONFIRM	<p>The parameter that limits confirmations to the receiving process only (YES/NO).</p>
XGRP_JRN_CTRL	<p>The parameter that allows JRN control messages to come from other groups (YES/NO).</p>
REDELIVERY_TIMER	<p>A value that specifies the time interval, in seconds, between attempts to deliver a recoverable message to a target that has exceeded its receive quota. This value is also used for redelivery attempts when a BEA MessageQ buffer pool is exhausted.</p> <p>Minimum = 0; maximum = 5000; default = 10 (seconds)</p>

PCJ_FILENAME	The default postconfirmation journal file (PCJ) name (64 characters). It is in effect until another name is set via a SET_PCJ message. The default value is DMQ\$MRS:MRS_ bbbb_ggggg .PCJ, where bbbb is the bus ID and ggggg is the group ID.
DLJ_FILENAME	The default DLJ filename (64 characters). It remains in effect until another name is set using a SET_PCJ message. The default value is DMQ\$MRS:MRS_ bbbb_ggggg .DLJ, where bbbb is the bus ID and ggggg is the group ID.

Note: After you set the MRS configuration parameters, you must restart the queuing group for these values to take effect. Use the `Restart/Shutdown COM_SERVER` option on the BEA MessageQ main menu to restart the group.

How BEA MessageQ Manages Destination Queue Files (DQFs)

Each permanent queue that receives recoverable messages requires a destination queue file located in the directory referred to by the logical name `DMQ$MRS`. DQF files are dynamically created and deleted as needed by the recovery system. Dynamic creation of DQF files provides the following benefits:

- No system management input is required to start receiving recoverable messages.
- You do not have to make any calculations of how much disk space to assign to each queue. Instead, systems are sized on the amount of disk space available, and how much of that space is to be allotted to recoverable messaging.

Note: After the pool of disk space that is allotted to recoverable messaging is exhausted, the recovery system forces the Undeliverable Message Action (UMA) to occur for each inbound message until the space becomes available.

The destination queue file is usually larger than the queue. Therefore, if the MRS Server wrote each message in the DQF to the target queue, it would overflow. To make sure that the MRS Server does not overflow the target queue, set the MRS parameter `CACHE_PERCENTAGE` to a value that is smaller than 100 and specify queue quotas for each queue that accepts recoverable messages.

Specifying the Location of MRS Files

Choose a common directory where the DQF files and SAF files will be written. You should consider which other files are used by your programs (database, logs, and so on) and project available space on the disk. The location is specified by the logical name `DMQ$MRS`. Define `DMQ$MRS` as follows in the `DMQ$BOOT` command procedure file:

```
$ define/table=DMQ$LNM_TBL/EXEC/nolog DMQ$MRS directory_spec
```

The default location is `DMQ$DISK:[DMQ$V50.MRS_bbbb_gggg]`, where `bbbb` is the 4-digit bus ID and `gggg` is the 5-digit group ID.

In cases where disk usage/loading is intense, BEA MessageQ supports a list of directories located on separate devices, for example:

```
$ define/table=DMQ$LNM_TBL/EXEC/nolog DMQ$MRS disk1:[mrmdir],  
-disk2:[mrmdir], diskn:[mrmdir]
```

Sizing MRS File Space

The MRS file system is divided into a variable number of fixed-sized areas. These areas are dynamically created before they are used and are assigned as needed.

Sizing the Amount of Recovery File Space

File creation occurs asynchronously before files are used, based on the information accumulated on past loads. These files are known as FREE files. The minimum and maximum FREE files used can be controlled via the logicals `DMQ$MRS_FREE_FILE_MIN` and `DMQ$MRS_FREE_FILE_MAX` in the

5 *Configuring Message Recovery*

DMQ\$SET_SERVER_LOGICALS.COM file in the DMQ\$USER area for a specific group. The default minimum FREE files is ten (10). To disable the use of FREE files, set the DMQ\$MRS_FREE_FILE_MIN to zero (0). The sender program should not usually experience delays due to dynamic file creation; however, under heavy load, delays may occasionally occur.

The total disk space taken up by journals can be controlled by the parameters listed in Table 5-2.

Table 5-2 Determining Journal File Size

Journal Parameter	Definition
AREA_SIZE	This parameter determines the size in disk blocks for each journal file. The default is 512. This size may need to be increased for performance reasons to cut down on the number of file creations required.
NUM_DQF_AREAS	This parameter determines the maximum number of Destination Queue files (DQF) that can be held by a group. Therefore, the maximum amount of the disk space taken by DQF files is equal to AREA_SIZE * NUM_DQF_AREAS. The minimum allowed is 100.
NUM_SAF_AREAS	This parameter determines the maximum number of Store and Forward (SAF) files that can be held by a group. Therefore, the maximum amount of the disk space taken by SAF files is equal to AREA_SIZE * NUM_SAF_AREAS. A setting of zero (0) will disable Store and Forward Journaling.
NUM_DLJ_AREAS	This parameter determines the maximum number of Dead Letter Journal (DLJ) files that can be held by a group. Therefore, the maximum amount of the disk space taken by DLJ files is equal to AREA_SIZE * NUM_DQF_AREAS. A setting of zero (0) will disable Dead Letter Journaling.
NUM_PCJ_AREAS	This parameter determines the maximum number of Post Confirmation Journal (PCJ) files that can be held by a group. Therefore, the maximum amount of the disk space taken by PCJ files is equal to AREA_SIZE * NUM_PCJ_AREAS. A setting of zero (0) will disable Post Confirmation Journaling.

Sizing the Journal File Chunk Size

The journal file chunk size defaults to 512 bytes. This value is controlled by the `DMQ$MRS_CHUNK_SIZE` logical in the `DMQ$SET_SERVER_LOGICALS.COM` file in the groups `DMQ$USER` area. This value can be set for performance tuning. A chunk size of 512 bytes can hold up to a 480 byte message including the message header. Each chunk has a 32 byte record header. It is recommended that the chunk size reflect the average message size, but cannot exceed 32767 bytes. The chunk size will be block aligned by MRS.

Sizing MRS Server In-Memory Data Structures at Startup

The `NUM_QUEUES` and `NUM_MESSAGES` fields are used to set the initial size of the key MRS data structures when the MRS Server starts up.

`NUM_QUEUES` sets the number of queues (both remote and local) that MRS can keep track of. `NUM_MESSAGES` sets the number of messages that MRS can track. Both these parameters expand dynamically as needed.

Both `NUM_QUEUES` and `NUM_MESSAGES` can be automatically sized to the largest values from previous runs based on the setting of the `USE_HIGH_WATER_MARK` parameter. This allows MRS to allocate space more efficiently than if it expands its data structures several times. You can control the sizing by setting these parameters in the MRS and JRN initialization table.

While the MRS Server is running, it might request additional space to hold messages until confirmed.

Note: If the value for maximum outgoing messages is too high, the MRS Server will not be able to allocate enough virtual memory without altering `PGFLQUO` for the MRS Server or the `SYSGEN` parameter `VIRTUALPAGECNT`. Large DQFs can also boost MRS memory allocation requests beyond `PGFLQUO`.

Reducing MRS Server Startup Time in Large-Scale Applications

Reading all recoverable messages and building an in-memory index are operations that do not scale well as the number of recoverable messages increases. To address this, recoverable files do not have to be scanned prior to use. This, however, means that reliable counts of the number of recoverable messages are not normally available immediately when the group starts up.

A switch setting is provided to enable accurate counts on startup, when accurate counts of messages are required prior to starting messaging.

Set `LOAD_MRS_CTRS` to `NO` to reduce startup time by not counting messages.

Set `LOAD_MRS_CTRS` to `YES` to have MRS count all messages prior to startup.

MRS Internal Operation Tracing at Startup

MRS tracing can be turned on or off and redirected dynamically. You should not enable MRS internal operation tracing unless instructed to do so by BEA Systems Customer Support. Tracing slows down the operation of the MRS Server and consumes a significant amount of disk space. Refer to Chapter 10, “Using BEA MessageQ System Management Utilities,” for detailed information on process and error tracing.

Confirming Message Removal from the Recovery System

In BEA MessageQ, messages are removed from the recovery system through implicit or explicit confirmation. To explicitly confirm messages, the receiver program uses the `pams_confirm_msg` service. The action of the `pams_confirm_msg` service is determined by:

- `CONFIRM_STYLE` queue attribute (from the queue configuration table (QCT))
- `DMQ$BLOCKING_CONFIRM` logical name
- `RCVR_ONLY_CONFIRM` parameter (from the MRS and JRN initialization table)

Using the `CONFIRM_STYLE` Attribute

The `CONFIRM_STYLE` attribute from the Queue Configuration Table (QCT) controls how messages can be confirmed. The QCT is described in Chapter 4, “Configuring Message Queues and Global Memory.” This section describes the `CONFIRM_STYLE` attribute only.

Valid `CONFIRM_STYLE` attribute settings are listed in Table 5-3.

Table 5-3 Valid `CONFIRM_STYLE` Attribute Settings

Setting	Description and Use
II	Implicit, In-order confirmation —If the confirm mode is set to [II], then each call to <code>pams_get_msg</code> or <code>pams_get_msgw</code> for an attached queue confirms the recoverable message from the previous call. Therefore, no special coding needs to be done in the application to correctly process recoverable messages. The messages are confirmed in the order they are dequeued. This is the most efficient way to confirm messages.

EI	Explicit, In-order confirmation —Explicit confirmation [EI] is useful when several messages need to be collected and then a single transaction applied using data from all the messages. The <code>pams_confirm_msg</code> service is used after all the information has been received and correctly applied. Setting the confirm mode to EI allows such a behavior. The messages must be confirmed in the order they are dequeued.
EO	Explicit, Out-of-order confirmation —This confirmation style is the same as explicit, in-order confirmation with the exception that several messages may be dequeued and then confirmed <i>in any order</i> by a call to <code>pams_confirm_msg</code> . This can be useful when handling multiple simultaneous transactions from several sources.

Note: A period (.) specifies the default which is EO.

Controlling Recovery System Response with DMQ\$BLOCKING_CONFIRM

The logical name `DMQ$BLOCKING_CONFIRM` is used to control whether or not `pams_confirm_msg` waits for a response from the recovery system.

`DMQ$BLOCKING_CONFIRM` can be set in the process table to YES or NO; the default is YES. The messaging rate for draining recoverable messages from a DQF journal can be increased significantly by setting the logical name `DMQ$BLOCKING_CONFIRM` to NO.

This setting causes `pams_confirm_msg` to release control to the application immediately after the confirmation request is made to the recovery system instead of blocking for a response that indicates that the message has been deleted.

Note: Set `DMQ$BLOCKING_CONFIRM` to NO only if the application can tolerate the occasional redelivery of a message in the case where a failure/restart occurs after a confirmation request is made, but before it is served.

`DMQ$BLOCKING_CONFIRM` can be set for a single process by defining the name in the process logical name table. It can also be set group-wide, by defining the name in `DMQ$LNM_TBL`.

Using RCVR_ONLY_CONFIRM Parameter

The `RCVR_ONLY_CONFIRM` parameter in the MRS initialization section of the group initialization file can be set (YES or NO) to limit message confirmations to the receiving process only.

6 Setting Up Selective Broadcasting

Selective broadcasting enables applications to send messages to multiple destinations using a single program call. This chapter discusses the following topics:

- Overview of Selective Broadcasting
- Starting the SBS Server
- Configuring Broadcasting
- SBS Interoperability
- Ethernet Broadcast Recovery Methods

Overview of Selective Broadcasting

BEA MessageQ Selective Broadcast Services (SBS) enable applications to send a message to many receiver programs using a single program call. The sender program uses the standard `pams_put_msg` service to send a message to a set of queues registered to receive broadcast messages. The SBS Server also maintains the queue availability database used by the AVAIL/UNAVAIL services.

BEA MessageQ for OpenVMS includes the following SBS features:

- Retransmission Protocol (RP) for Ethernet multicasting—Each universal Multipoint Outbound Target (MOT) that supports Ethernet multicasting can be configured with an option to retry transmission of cross-group messages in the

event of delivery problems. Messages sent to a MOT that is configured for retransmission are stored in the SBS retransmission list after they are broadcast. The size of the retransmission list is configured in the %SBS section of the group initialization file. These parameter sets the maximum number of messages stored by the SBS Server to fulfill retransmission requests in the event of message delivery failures. See “Ethernet Broadcast Recovery Methods” for additional information.

- **Message API**—The RISC aligned message API includes rule based broadcast selection criteria and status reporting.
- **Large message support**—SBS contains support for large (4 MB) messages using the standard message transport (TCP/IP or DECnet), however direct Ethernet multicasting is limited to a maximum message size of 32000 bytes.
- **Communication service selection**—SBS introduces the `COMM_SERVICE` to the %SBS section of the `DMQ$INIT` file. The `COMM_SERVICE` is intended to prioritize and support the future segmentation of broadcast streams.
- **Dual rail Ethernet configuration**—SBS supports dual rail Ethernet configurations for greater broadcast reliability.

Note: In version 4.0A and version 5.0 of BEA MessageQ for OpenVMS, SBS includes a set of message-based services which obsoletes the message interfaces used in version 3.2B and prior versions. However, applications that currently use the prior versions of SBS will continue to run in a version 4.0A or version 5.0 BEA MessageQ for OpenVMS environment. The older, obsolete, message interfaces are not supported on non-OpenVMS SBS platforms such as UNIX or Windows NT.

Starting the SBS Server

The BEA MessageQ SBS Server must be running for applications to use either the Selective Broadcast or AVAIL/UNAVAIL services. To start the SBS Server for a message queuing group, edit the Profile section of the `DMQ$INIT.TXT` file and set the symbol `ENABLE_SBS` set to `YES`, and then start the message queuing group. The

ENABLE_SBS parameter is set to YES by default in the group initialization file template supplied with BEA MessageQ for OpenVMS. Listing 6-1 shows the ENABLE_SBS parameter in the Profile section of the group initialization file.

Listing 6-1 SBS Enable Parameter

```
%PROFILE          ***** Profile Parameters *****
*
. . .
ENABLE_MRS      YES   ! Enable MessageQ Message Recovery Services
ENABLE_JRN      YES   ! Enable MessageQ Message Journaling Services
ENABLE_SBS      YES   ! Enable MessageQ Selective Broadcast Services
. . .
*
%EOS
```

Configuring Broadcasting

Use the SBS Server initialization section of `DMQ$INIT.TXT` to configure the broadcasting operation of the local SBS Server. This section of `DMQ$INIT.TXT` begins with `%SBS`.

This section covers the following information:

- Specifying Multipoint Outbound Target (MOT) Addresses
- Configuring Optimized Ethernet Mode
- Broadcasting Requirements and Restrictions

Specifying Multipoint Outbound Target (MOT) Addresses

A Multipoint Outbound Target (MOT) is a queue address that is associated with a broadcast stream capable of broadcasting a single message to multiple target queues. MOT addresses are divided into two ranges, `PRIVATE` and `UNIVERSAL`. Messages sent

to PRIVATE MOTs are broadcast only within the local group; they are not transmitted to other SBS servers. Messages sent to UNIVERSAL MOTs are broadcast to the local group as well as to all SBS servers connected to the BEA MessageQ bus. For UNIVERSAL MOTs, a single broadcast message is sent to each remote SBS, then each SBS server broadcasts them locally to all interested target queues.

The PRIVATE range includes addresses from MOT_PRIVATE to MOT_RESERVED-1. BEA MessageQ broadcasts messages within the PRIVATE range using a list maintained by the local SBS Server.

The UNIVERSAL range includes addresses from MOT_RESERVED to MOT_UNIVERSAL. BEA MessageQ broadcasts messages within the UNIVERSAL range to all SBS servers on the message queuing bus and then each SBS server broadcasts them locally to all interested target queues.

MOT_PRIVATE, MOT_RESERVED, and MOT_UNIVERSAL are defined in Table 6-1

Table 6-1 %SBS MOT Address Ranges

Parameter	Address	Description
MOT_PRIVATE	4000	Private MOT address are in the range of 4000-4900, inclusive. Messages broadcast to these address will remain local to the queuing group.
MOT_RESERVED	5000	MOT address settings 4901 to 5099 are reserved for BEA MessageQ.
MOT_UNIVERSAL	6000	Universal MOT address are in the range of 5100-5999, inclusive. Messages broadcast to these addresses will be transmitted outside the queuing group.

Note: MOT ranges 4901 to 4999 and 5000 to 5099 are reserved for BEA MessageQ.

Configuring Optimized Ethernet Mode

The optimized Ethernet mode on the SBS server utilizes the hardware multicasting available with Ethernet controllers. Therefore, instead of simulating broadcasting by sending a message to each adjacent SBS server over the standard BEA MessageQ

COMM_SERVICE	<p>A two-field record. The first field is a unique integer between 0 and 10 that indicates which communication service is being defined. (Starting with BEA MessageQ v4.0A, the only services available are 0-10.) The COMM_SERVICE number is intended to be used to control the priority of one path over another when multiple paths to a group are detected. The lower the number the higher the priority. Services 1-9 are optional and correspond to the old DMQ\$INIT Ethernet SET command numbers. Service 10 is reserved for the default unoptimized BEA MessageQ broadcasting, must be set to the DMQ transport at this time. The second field indicates the method of server-to-server communications this service uses. The first part is the protocol type and the second is the transport type. The two parts must be separated with a / (slash) character.</p>
Protocols	<ul style="list-style-type: none">■ DG—Datagram protocol. Messages are sent "best try" with failures detected by sequence gaps.■ DG_OLD—Datagram protocol. Same as DG but uses the older protocol used in BEA MessageQ for OpenVMS version 3.2 and earlier releases. Only valid in conjunction with the ETH transport.■ RP—Retransmit Protocol. Messages are sent "best try" with failures recovered via retransmit request protocol up to a depth of messages that can be set by the user.
Transports	<ul style="list-style-type: none">■ DMQ—BEA MessageQ messaging. Messages are sent using BEA MessageQ as the transport medium.■ ETH—Direct Ethernet. Messages are sent using the hardware multicast capabilities of the Ethernet controller.
GROUPS	<p>A numeric field that contains a single BEA MessageQ group number per line. Used to indicate which groups should be broadcasted to. Starting with BEA MessageQ version 4.0A the required setting is an asterisk (*), used to indicate all known groups. This keyword is only useful when BEA MessageQ is used as a transport. The default is *.</p>
DEVICE_1 DEVICE_2	<p>A text field that specifies the name of the Ethernet controller to use for each rail. Single-rail configurations use the DEVICE_1 keyword only; dual-rail configurations use both. This keyword is useful only in conjunction with the Ethernet transport. In a given initialization file, only one setting may be entered for DEVICE_1/DEVICE_2. If multiple settings are entered, only the last will be used.</p>
DRIVER_BUFFERS	<p>A numeric field that contains the number of buffers that the system Ethernet controller should use to buffer incoming packets. On OpenVMS, this directly affects the amount of non-paged pool allocated to each channel by the Ethernet controller. The default is 16. In a given initialization file, only one setting may be entered. If multiple settings are entered, only the last will be used.</p>

6 *Setting Up Selective Broadcasting*

CNTRL_CHAN	A three-field record that specifies the Ethernet protocol/address pair for control information. The first field identifies the Ethernet hexadecimal multicast address (the low-order bit of the first byte must be set) that the SBS Server uses in the Ethernet transport mode. The second hexadecimal field is a unique protocol number used by the SBS Servers. The address/protocol pair for the SBS control channel must be consistent in all cooperating SBS Servers and must be distinguishable from other address/protocol pairs (for example, LAT, DECnet, local area VMScusters, and so on).
DATA_CHAN	A two-field record that specifies the data channel for optimized communications. The address/protocol pair for the SBS data channel must be consistent in all cooperating SBS Servers and must be distinguishable from other address/protocol pairs (for example, LAT, DECnet, local area VMScusters, and so on). The first field is the hexadecimal Ethernet multicast address in the form of XX-XX-XX-XX-XX-XX. The second field is the hexadecimal Ethernet protocol number.
REGISTER	A one-field or six-field record used to add a subscription ID (MOT) to the communication service. The number of fields depends on the type of service being specified. For DG/DMQ it is a one-field record; all others are six-field.
MOT	Subscription ID that identifies the unique broadcast stream. For datagram protocols using the DMQ transport, a wildcard character (*) can be used to indicate all possible subscription IDs. Make sure you record the number of MABs a given Subscription ID has because it has direct effect on the total amount of virtual memory that the SBS Server uses.
Transmit Silo	Number of Message Assembly Buffers (MABs) used to manage retransmission requests. For Datagram protocols this will always be forced to one. Valid ranges 0 to 100.
Receive Silo	Number of MABs to use for simultaneous multi-packet receptions. Valid ranges 0 to 100. For DG_OLD protocols this will always be forced to 5.
Maximum Heartbeat	Number of unacknowledged Polls before a sender is considered dead. Valid ranges 0 to 100.
Poll Interval	Number of MOT heartbeat intervals without an intervening message before a poll request is generated. Valid range is 0 to 100.
Dead Poll Interval	Used as a multiplier to the Poll Interval to reduce the number of Polls sent to an unresponsive or inactive group. Valid range is 0 to 100.
END_COMM_SERVICE	Keyword used to close a communication service description block.

Listing 6-3 shows the Ethernet Broadcasting Parameter Settings in the Queue Configuration Table (QCT) section of the group initialization file.

Listing 6-3 Ethernet Broadcasting Parameters in %QCT Initialization File Section

```
%QCT          ***** Queue Configuration Table *****
*
*
*          ---Pool Quota---  UCB  Q  Q  Confrm Perm Name  Chk
* Queue Name      Num  Bytes  Msgs Ctrl  Send Type Own  Style Act  Scope ACL
*-----
*
*
*
* SBS Server uses the following UCB numbers for Optimized Delivery
*
SBS_ETH_CONTROL      74      0  0  .  E  .  .  .  .  L  N
SBS_ETH_CHAN1        75      0  0  .  E  .  .  .  .  L  N
SBS_ETH_CHAN2        76      0  0  .  E  .  .  .  .  L  N
*
*
*
*
*EOS
```

The queue configuration table (%QCT) section of the DMQ\$INIT.TXT file must contain entries for the CNTRL_CHAN and each of the DATA_CHAN entries. These entries must correspond to the correct queue address, have zero quota, and have the E flag set in the UCB_Send field.

Broadcasting Requirements and Restrictions

A registry of reserved protocols and multicast addresses is maintained by the Xerox Corporation. Choose protocols and multicast addresses so that they do not conflict with other users of the local area network (LAN). See Chapter 9 of the *OpenVMS I/O User's Reference Manual*.

The following are requirements and restrictions imposed on SBS:

- BEA MessageQ does not restrict the range of addresses in any way. However, they must be multicast addresses (which are defined as having the low bit of the first byte set).

- Multicast receiver addresses are a limited resource maintained by the Ethernet controller. Most Ethernet controllers allow 11 to 13 simultaneous multicast addresses, however other network protocols share these hardware limited resources.
- Ethernet protocol numbers are maintained at the driver level and must be unique within the system unit, regardless of what the Ethernet address is set to. For example, if NI OpenVMS clusters are enabled (which use the protocol 60-07), the SBS server cannot use the protocol 60-07.
- The SBS server can be started anytime on an Ethernet controller. However, if the SBS server is sharing an Ethernet controller with DECnet, it should be started after DECnet and TCP stacks. This allows DECnet/TCPIP to set network device characteristics, which an SBS server startup would prevent.

SBS Interoperability

The following list describes the supported interoperability between versions and platforms running the SBS Server in BEA MessageQ for OpenVMS V5.0:

- OpenVMS version 5.0 to OpenVMS version 4.0A—full interoperability over both BEA MessageQ and Ethernet transports. The Ethernet transport supports either datagram or retransmission protocols.
- OpenVMS version 5.0 to Non-OpenVMS version 4.0A—interoperability over the BEA MessageQ transport.
- OpenVMS version 5.0 to OpenVMS Pre-version 4.0A—interoperability over the BEA MessageQ or Ethernet transports. Ethernet transport supports only datagram protocol.
- Non-OpenVMS version 5.0 to OpenVMS Pre-version 4.0A—not supported.

Ethernet Broadcast Recovery Methods

The BEA MessageQ for OpenVMS SBS Server incorporates an Ethernet retransmission protocol which allows the automatic recovery of lost message packets. A receiving SBS Server can request lost message packet retransmissions up to a selectable depth of message packets.

This feature allows the SBS Server to attempt to recover lost messages identified by a sequence gap. This feature, combined with dual-rail support, significantly enhances the reliability of optimized Ethernet broadcasting.

Configuring Optimized Ethernet Mode

The optimized Ethernet mode on the SBS Server uses the hardware multicasting available with Ethernet controllers. Instead of simulating broadcasting by sending a message to each adjacent SBS Server over the standard BEA MessageQ links, it broadcasts one message to all listening systems on the bus. This mode has the advantage of scalability: a single copy of a message is transmitted across the network and is received only by those systems which have applications registered for the message's MOT. Therefore, this facility improves network utilization by limiting the traffic per message placed on the network.

Because messages sent via direct Ethernet access are limited to 1,500 bytes, the sending and receiving SBS Servers handle segment and reassemble the messages. This segmentation, coupled with network load, increases the probability that a packet collision may result in a lost packet at one or more receiver systems. A lost packet, in turn, results in a lost message, which is reported to receiving applications as a broadcast sequence gap.

Note: Although BEA MessageQ supports message sizes up to 4MB, optimized Ethernet mode limits message sizes to 32,000 bytes.

In the following discussion, refer to Listing 6-2, which shows the SBS Server initialization section of the group initialization file template, as well as Table 6-2, which describes parameters and fields.

Dual-rail Mode

BEA MessageQ provides two features to increase the reliability of messages sent using Optimized Ethernet Mode, dual-rail mode and the RP/ETH retransmission protocol. With dual-rail mode, SBS supports up to two (2) Ethernet devices per system. Using two Ethernet devices decreases the chance that a lost packet resulting from a network collision will result in a lost message. The sending system writes a copy of the message to the devices specified in the `DEVICE_1` and `DEVICE_2` parameters for `COMM_SERVICE` with a value of `DG_OLD`. Receiving SBS Servers construct a single copy of the message from packets received from both Ethernet devices. If a packet contained in a particular message is not received from either rail, a sequence gap message is sent to receiving applications that requested sequence gap notification.

To configure dual-rail mode, specify the `Prot/Xport` field of the `COMM_SERVICE` as `DG_OLD`. Specify valid device addresses for both Ethernet devices on your systems, insuring that you specify device unit 0 (for example, `ECA0:` or `EWB0:`). Ensure that the Ethernet devices specified in the `DEVICE_1` and `DEVICE_2` fields of various groups are actually connected on the same network. The simplest method is to build single-rail configurations for all groups (specify only `DEVICE_1`) and use the `DMQ$EXE:DMQ$SBS_EXAMPLE` program to verify broadcast reception in each group. Repeat this test using the devices specified for `DEVICE_2` as `DEVICE_1` to ensure that the final dual-rail configuration is properly configured.

The only field in the `REGISTER` line that is specified by the user with this service is the `MOT`. The transmit and receive silos are forced to their default values of 1 and 5, respectively; the heartbeat parameters do not affect this service.

RP/ETH Retransmission Protocol

There are some applications where broadcast messages need a higher probability of delivery than is provided by simple dual-rail redundancy. SBS provides the RP/ETH retransmission protocol by which a receiving SBS Server that cannot construct a complete message may request it from the sending SBS Server. If the sender still has the message in its cache, it will resend it to requesting system.

It is important to understand that the retransmission protocol is not a recoverable delivery protocol. That is, messages sent to a MOT whose `COMM_SERVICE` is the retransmission protocol (RP/ETH) may still result in sequence gap at one or more receiver systems. This protocol is best suited to broadcast traffic with a predictable delivery rate.

The RP/ETH communication service operates as follows. Any message sent to a MOT whose service is RP/ETH are placed in a cache called the *transmit silo* after they are sent. The transmit silo size is measured in Message Assembly Buffers (MABs). Each MAB contains a complete copy of the largest message (up to 32,000 bytes) that this queuing group can send or receive. MABs also contain sequencing information used to control retransmission requests. Transmit silos are first-in/first-out: the oldest message in the transmit silo is removed when a new message requires storage. The transmit silo size is specified by the Transmit Silo parameter in the `REGISTER` statement used to define the MOT.

If a missing segment is detected by a receiving SBS Server, it requests retransmission of the message from the point at which the first missing segment was detected. This request is sent via a high-priority message (transmitted via the standard BEA MessageQ cross-group transport, either TCP/IP or DECnet) to the sending SBS server. The requested message fragment (or an entire message) is returned via a high-priority message, which is also transmitted via the standard BEA MessageQ cross-group transport. If the message has already been flushed from the sending group's transmit silo, a sequence gap notification is delivered to the receiving applications.

Each MOT using RP/ETH also has a receiver cache called the *receive silo*. The receive silo accumulates messages to ensure in-order delivery to registered applications. For example, a sender broadcasts messages 46, 47 and 48 to an RP/ETH MOT but the receiver receives only messages 46 and 48. It is necessary for the receiving system to hold message 48 until the retransmitted copy of message 47 is received. Receive silo size is also measured in MABs. The receive silo size is specified by the Receive Silo parameter in the `REGISTER` statement used to define the MOT.

RP/ETH Receive Silo Rules

Receive silos are emptied according to the following rules:

1. The receive silo is emptied as soon as possible. In the example described in "RP/ETH Retransmission Protocol," message 46 is delivered as soon as it is complete, regardless of the size of the receive silo, or the fact that message 48, instead of 47, is received next.

2. Missing messages cause MABs to be reserved for them in anticipation of their retransmission. This increases the chance that a message sent by a (relatively costly) retransmission operation will be delivered to applications. This delivery, however, is not guaranteed, as explained in rule 3.
3. If the receive silo has exactly one MAB remaining, filling that MAB with a complete message will cause the silo to be emptied immediately (regardless of the state of pending retransmission requests). Appropriate sequence gap notifications will occur in this case.
4. A sequence of missing messages which causes the receive silo to be filled immediately causes a sequence gap to be generated for the entire sequence. For example, assume that the receive silo size is 3, the last message received is 22, and the silo is empty. If the next message received is 28, a five-message sequence gap (representing messages 23-27) is generated followed by delivery of message 28.
5. Receive silos are logically segmented by sending group. Silos are emptied on a per-sending-group basis. That is, a completed message from group 14 is delivered regardless of the state of messages from groups in this MOT's receive silo. First-in/first-out ordering is thus preserved on a sending group basis.

RP/ETH Examples

SBS Servers using optimized Ethernet provide timely detection of missing messages. Consider the case where a sending group broadcasts message 26, which is short enough to be contained in a single packet. Assume that a receiving group does not receive this packet due to a network collision. The receiving group does not detect that it missed message 26 until message 27 is broadcast. To prevent this condition, SBS Servers using RP/ETH protocol exchange status inquiries on a periodic basis. The minimum interval between these inquiries is called the heartbeat and is specified in 1 millisecond units. Thus, if the heartbeat parameter were set to 1000, status polling exchange occurs no more than once per second. The `HEARTBEAT` statement specifies this value for a particular group.

Recognizing that MOTs may vary with respect to their expected traffic rates, BEA MessageQ allows you to control the polling on a per-MOT basis. It does so with three parameters specified for the MOT in a `REGISTER` statement: maximum heartbeat, poll interval and dead poll interval. These values are multiples of the heartbeat specified for the group. Thus, if the heartbeat were set to 2000 (that is, 2 seconds) and the poll

interval were set to 2, the actions associated with the poll interval would be executed after 4 seconds had elapsed. The following example shows how these parameters interact.

Assume that you have 2 groups, 30 and 31, using RP/ETH for MOT 5101. To simplify things, assume that group 30 only sends broadcast messages and group 31 only receives these messages. The relevant parameters in the SBS section in group 31's initialization file are:

```
HEARTBEAT 1000
COMM_SERVICE 0 RP/ETH
!           Transmit silo  Recv silo  Max HB  Poll  Dead Poll
REGISTER 5101           10           10       4     5     10
```

The following sequence of operations occurs:

1. An application in group 21 registers to receive message on MOT 5101.
2. After registration is complete, an application in group 20 sends a message to MOT 5101. Group 30's SBS Server uses optimized Ethernet to send this message.
3. Group 31 receives this message and distributes it to the registered application. Upon receipt, the poll interval timer is set for this MOT and group. If a message is received from group 30 for MOT 5101 before the poll interval has elapsed (in this case, $5 * 1000\text{msec} = 5$ seconds), then the poll interval timer is reset.
4. If no messages are received in the poll interval, group 21 sends a poll request to group 30 to determine the sequence number of the last message sent on MOT 5101.
5. Group 31 has the maximum heartbeat chances to respond. This equates to 20 seconds ($4 * 5 * 1000\text{msec} = 20$ seconds).
6. Group 31 may respond with the sequence number of the last message received by group 31 for MOT 5101. If this is the case, no corrective action needs to be taken because there hasn't been any further traffic from group 30 on this MOT.
7. Group 31 may respond with a larger sequence number than the last message received by group 31 for MOT 5101. In this case, group 30 attempts to refill its receive silo in accordance with the rules specified above.

8. If 20 seconds elapse without a response (or group 21 is detected to be down by group 20), then group 30 is assumed to be dead. In this case, the polling rate is reduced to the dead poll interval of 50 seconds ($5 [\text{poll interval}] * 10 [\text{dead poll interval}] * 1000\text{msec} = 50 \text{ seconds}$).
9. When group 31 is restarted and traffic begins to arrive for MOT 5101, the polling is reset to the poll interval as specified in step 3.

RP/ETH Configuration

Using RP/ETH protocol is most effective if you limit its use to a small number of MOTs that have predictable traffic patterns. Specify the `Prot/Xport` field of the `COMM_SERVICE` as `RP/ETH`. Specify valid device addresses for both Ethernet devices on your systems, ensuring that you specify device unit 0 (for example, `ECA0:` or `EWB0:`). Ensure that the Ethernet devices specified in the `DEVICE_1` and `DEVICE_2` fields of various groups are actually connected on the same network. The simplest method for ensuring this is to build single-rail configurations for all groups (that is, just specify `DEVICE_1`) and use the `DMQ$EXE:DMQ$SBS_EXAMPLE` program to verify broadcast reception in each group. Repeat this test using the devices specified for `DEVICE_2` as `DEVICE_1` to ensure that the final dual-rail configuration is properly configured.

Set all parameters on the `REGISTER` line for each MOT to `RP/ETH`. Sending groups should specify more Transmit Silo MABs, and Receiving Groups should specify more Receive Silo MABs, thereby increasing the likelihood of retransmission requests being fulfilled. Note that each MAB requires a buffer equal to the size of the group's maximum message (up to 32,000 bytes) plus 150 bytes, so you should expect to increase the `PGFLQUO` parameter in `SBS` Section of `DMQ$USER:SET_SERVER_QUOTAS.COM` to reflect this increased virtual memory demand.

Tuning this service is an iterative process. There are message-based API calls which allow you to retrieve statistics for traffic and retransmission activity by MOT. See the `sbs_status_req` and `sbs_status_resp` messages in the *BEA MessageQ Programmer's Guide* for more information.

7 Creating Global Names

The BEA MessageQ application programming interface uses a numeric queue address to refer to a specific message queue within a message queuing group. The BEA MessageQ naming feature enables a name to be associated with a numeric queue address to separate the application from the underlying network configuration. Local names are shared only by applications that run in the same message queuing group; global names can be used by any application on the message queuing bus. BEA MessageQ includes the ability to define both local and global names.

To use local (group-wide) naming, configure queue names in the Queue Configuration Table (%QCT) or the Group Name Table (%GNT) section of the group initialization file. When the group starts up, BEA MessageQ automatically creates the group namespace. It creates the process namespace when an application attaches to the message queuing bus.

To enable your application to use global (bus-wide) naming, you must perform additional configuration steps. This chapter explains how to configure the BEA MessageQ built-in global naming capability and how to use Distributed Name Services (DNS).

Configuring BEA MessageQ Global Naming

The first step in configuring global naming is to decide the group or groups in which the Naming Agent will run. BEA MessageQ allows you to specify a main group and an alternate group to run the Naming Agent. To configure a group to run the Naming Agent follow the steps outlined in “Configure Groups to Run or Use the Naming Agent.”

Configure Groups to Run or Use the Naming Agent

The BEA MessageQ Naming Agent is the BEA MessageQ Server that maintains the namespace for name-to-queue address translations and performs the run-time queue lookup when an application refers to a queue by name. The %NAM section of the group initialization file allows the user to define a primary and an alternate Naming Agent group. BEA MessageQ allow the definition of up to two Naming Agents for each message queuing bus.

When BEA MessageQ starts each group, it looks in this section of the initialization file to decide whether to start a Naming Agent for the group. For groups that do not run a Naming Agent, BEA MessageQ uses the information in the %NAM section to direct requests to the Naming Agent. Groups must have a cross-group connections to the groups in which the Naming Agent runs. Listing 7-1 shows a sample %NAM section.

Listing 7-1 Sample%NAM Section

```
%NAM ***** Naming Agent Section *****
* This section consists of a maximum of 2 entries consisting of
* a keyword, "NA_GROUP", followed by the group number of a group
* where a naming agent is running
%NAM
NA_GROUP    10
NA_GROUP    28
%EOS
```

Valid ranges for the NA_GROUP parameter are 0 to 32,000. A value of 0 indicates that a Naming Agent should be started in the local group. Note that any Naming Agent startup failures will be logged in DMQ\$LOG:DMQ\$NA_SERVER_bbbb_gggg.LOG, not the group's Event Log.

Configure a Lightweight Namespace

BEA MessageQ for OpenVMS supports the creation of both a lightweight namespace, which is included with BEA MessageQ, and a DNS namespace. To create the lightweight namespace, BEA MessageQ uses a flat file system by creating the

directory in which the BEA MessageQ Naming Agent will maintain the namespace. On BEA MessageQ for OpenVMS the following default lightweight namespace directory is created during installation:

```
DMQ$DISK : [ DMQ$V50 . DMQNS ]
```

To use global naming, you must create a namespace on the nodes on which the Naming Agents will run. BEA MessageQ enables users to configure two Naming Agents to support global messaging for the environment. In order to allow the second Naming Agent to form a backup for the first, both Naming Agents must be configured to use the same namespace. Therefore, when you configure your namespace for use by two Naming Agents that run on different systems, it must use a shared file system such that is accessible to both Naming Agents.

After you create the namespace, you must set the `DMQNS_DEVICE` environment variable to specify a device name for the namespace because access to the BEA MessageQ lightweight namespace for global naming is system dependent. Therefore, when a Naming Agent is configured, it must be told what device name to use when it accesses this namespace. This is done by setting the environment variable `DMQNS_DEVICE` as follows:

- For Windows NT, set the variable to a drive letter followed by a colon (for example, `C:`) or a fully qualified pathname (for example, `\\machine\share`).
- For OpenVMS, `DMQNS_DEVICE` is a logical (defined in `DMQ$BOOT.COM`), it should be set to the root device and/or directory for the namespace path. The default is `DMQ$DISK : [DMQ$V50]`.
- For UNIX, set the variable to a file system specification (for example, `/usr` or `/mnt/dmqns`).

Note that this environment variable need only be set for the group or groups in which the Naming Agent is running. Only the Naming Agent process is designed to use this environment variable setting to resolve the location of the namespace.

For environments that use two Naming Agents, it is critically important to ensure that the device name set using the `DMQNS_DEVICE` environment variable on both systems points to the same device that stores the shared file system containing the BEA MessageQ namespace.

Default Namespace Path Definition

BEA MessageQ uses the `DMQNS_DEFAULTPATH` logical name to define path information for the namespace. This logical name is defined in `DMQ$BOOT.COM`. Using the `DMQNS_DEFAULTPATH` logical name allows you to remove the `DEFAULT_NAMESPACE_PATH` definition from the `DMQ$INIT.TXT` file, yet still define the path information needed by the Naming Agent Server.

The Naming Agent Server only uses the `DMQNS_DEFAULTPATH` logical if `DEFAULT_NAMESPACE_PATH` is not defined in the `DMQ$INIT.TXT` file. (The `DEFAULT_NAMESPACE_PATH` parameter is commented out in the `DMQ$INIT.TXT` file provided in the `[DMQ$V50.USER.TEMPLATE]` subdirectory.) If the `DEFAULT_NAMESPACE_PATH` is defined, then the `DMQNS_DEFAULTPATH` logical is ignored.

Configure a Default Namespace Path for Each Group

To use a global name, specify at least some portion of the path name. Path information can be supplied by the application, either use the `DMQNS_DEFAULTPATH` logical name, or use the `DEFAULT_NAMESPACE_PATH` parameter in the `%PROFILE` section of the group initialization file to create and maintain path information for global names. For global naming to function properly, this parameter must be set to the same value for all groups in which applications are designed to access the same namespace.

To define a default namespace path as a directory called `/DMQNS/mydir` using the `DMQNS_DEFAULTPATH` logical name, enter the following command:

```
$ define DMQNS_DEFAULTPATH /DMQNS/mydir
```

The following syntax shows how to use the `DEFAULT_NAMESPACE_PATH` parameter in the `%PROFILE` section of the group initialization file to set the default namespace to be created and maintained in the directory called `/DMQNS/mydir`.

```
%PROFILE          ***** Profile Parameters *****  
*  
...  
DEFAULT_NAMESPACE_PATH /DMQNS/mydir/  
*  
%eos
```

Note: The `DEFAULT_NAMESPACE_PATH` is case sensitive. This is important to note for VMS groups that are referencing a namespace located on a Unix system.

For testing purposes, you might set this parameter to look at a copy of the production namespace that you store in your own development directory. However, when the application is deployed into production, the application will reference the common namespace shared by all production systems.

Define the Queue Names in the Group Initialization File

Use the Queue Configuration Table (`%QCT`) or the Group Name Table (`%GNT`) of the group initialization file to create static or dynamic definitions for global names as follows:

- Define global static names in the `%QCT` or `%GNT` by providing the name, the queue address, and setting the name scope identifier to `G` for global names.
- Define global dynamic names by supplying the name, `0.0` as the address and the `G` identifier for global names. Names defined with a `0.0` address can be dynamically bound to a queue address at run-time using the `pams_bind_q` function.

Listing 7-2 shows static and dynamic global name definitions in the `%GNT` section of the initialization file.

Listing 7-2 Sample Group Name Table for Global Naming

```
%GNT ***** Group Name Table Section *****
*
*      Queue Name                Group.Queue      Scope
*
-----
widgets                9.10            G
red_widgets            0.0             G
*
%EOS
```

When an application refers to a queue by name using the `pams_locate_q` or the `pams_bind_q` functions, it can specify the name as one of the following:

- **unqualified name**—The application uses only the queue name such as `widgets` and does not specify the path. The Naming Agent automatically assigns a prefix to the name with the value of the environment variable `DMQNS_DEVICE`. For example, if the `DMQNS_DEVICE` logical is set to `DUA0:` and the `DEFAULT_NAMESPACE_PATH` is set to `/inventory`, the Naming Agent would search for the name `widgets` in:

```
/DUA0/inventory/inventory.dnf
```

or

```
DUA0:[inventory]inventory.dnf
```

- **partially qualified name**—The application specifies the queue name and a portion of the path name. The Naming Agent automatically assigns a prefix to the pathname and queue name with the device specified as the `DMQNS_DEVICE` environment variable and the setting of the `DEFAULT_NAMESPACE_PATH` parameter. For example, if the `DMQNS_DEVICE` logical is set to `DUA0:` and the `DEFAULT_NAMESPACE_PATH` is set to `/inventory`, the Naming Agent would search for the name `test/widgets` in:

```
/DUA0/inventory/test/test.dnf
```

or

```
DUA0:[inventory.test]test.dnf
```

- **fully qualified name**—The application specifies that the name is a fully qualified name using a forward slash (`/`) as the first character of the name. When the first character of a name begins with a forward slash (`/`), the Naming Agent does not assign a prefix to any information to the name other than the device name specified by the `DMQNS_DEVICE` environment variable. This means that a fully qualified name includes the full pathname and queue name. For example, if the `DMQNS_DEVICE` logical is set to `DUA0:` and the `DEFAULT_NAMESPACE_PATH` is set to `/inventory`, the Naming Agent would search for the name `/production/test/widgets` in:

```
/DUA0/production/test/test.dnf
```

or

```
DUA0:[production.test]test.dnf
```

The use of unqualified, partially qualified, and fully qualified names gives application developers significant flexibility in using global name references.

Table 7-1 provides several more examples of how global names are resolved. In this example, the `DMQNS_DEVICE` environment variable is set to `DUA0:[DMQ.DMQNS]`.

Table 7-1 Sample Global Names and Their Resolution

Name Used in API	DEFAULT_NAMESPACE_PATH	Name Searched
toto	bus1	/DUA0/DMQ/DMQNS/bus1/toto
mypath/toto	bus1	/DUA0/DMQ/DMQNS/bus1/mypath/toto
/otherpath/toto	bus1	/DUA0/DMQ/DMQNS/otherpath/toto

Refer to the *BEA MessageQ Programmer's Guide* for more information on designing applications to use the BEA MessageQ global naming feature.

Using DNS with Global Naming

To use Distributed Name Service (DNS) naming with BEA MessageQ you must:

- Verify the presence of a DNS server on your network.
- Set your `DEFAULT_NAMESPACE_PATH` in the group initialization file to point to the DNS root directory. For example, if the node name is `NODE1`, the clearinghouse is `NODE1_CH`, the namespace is `NODE1_NS`, and the directory is `DMQ`, then the `DEFAULT_NAMESPACE_PATH` would be `NODE1_NS:.DMQ`.

Global Names that would generally be loaded into the lightweight namespace are loaded into the DNS namespace. The Manager Utility, `DMQ$MGR_UTILITY`, can be used to load names into the DNS namespace, and to browse this namespace. `DMQ$MGR_UTILITY` supersedes the `DMQ$NA_MGR.EXE` program provided in previous releases of BEA MessageQ.

The Manager utility updates the address of any name that has the `G` attribute in the `%GNT` section or `%QCT` section of the `DMQ$INIT.TXT` file, unless is dynamically bound to a particular queue in the group. Dynamically bound names are initialized to group 0 queue 0 and are modified by a call to `pams_bind_q`. To use *dynamic binding*, declare the names in the group where the dynamic binding will occur. This operational step prevents another group's startup from reinitializing the names.

Managing the Global Namespace

You can browse the global namespace, created names, and set and cleared addresses with the BEA MessageQ Manager utility program, `DMQ$MGR_UTILITY`. (The capabilities of the Naming Agent Manager utility have been incorporated into the Manager utility.) For information on using the Manager utility, see Chapter 10, “Using BEA MessageQ System Management Utilities.”

Viewing a Group’s Cache

The `DG` command (Display GROUP name table) in the main menu of the Manager utility (`DMQ$MGR_UTILITY.EXE`) allows the user to view all local and cached global names. See Listing 7-3 for a sample group name table.

Listing 7-3 DMQ\$MGR_UTILITY Display Group Name Table

```
Bus:99      Group:40      MessageQ Manager Utility      Thu Jan  8 12:47:48 1998
                                     Group Name Table

Entry  Scope    ID          Name
  1    G      40.00001   /DMQNS/GLOBAL_TEST
  2    G      40.00005   /DMQNS/GLOBAL_TEST_QUEUE5
  3    G      40.00006   /DMQNS/GLOBAL_TEST_QUEUE6
  4    G      40.00002   /DMQNS/QUEUE_2
  5    L      40.00091   ALL_UCBS
  6    L      40.00099   AVAIL_SERVER
  7    L      40.00100   COM_SERVER
  8    L      40.00100   CONNECT_SERVER
  9    L      40.00151   DCL_BY_Q_NAME
 10   L      40.00096   DEAD_LETTER_QUEUE
 11   L      40.00100   DECLARE_SERVER
 12   L      40.00153   DECNET_LD
 13   L      40.00191   DMQ_FULLLTEST_PQ
 14   L      40.00192   DMQ_FULLLTEST_SQ
 15   L      40.00150   DMQ_LOADER

[More]
<CR> to continue:
```

Global Name Examples

This section provides some examples of global name caching on BEA MessageQ for OpenVMS. These examples use `pams_locate_q` with the following information defined in the group initialization file (`DMQ$INIT.TXT`):

```
%PROFILE
DEFAULT_NAMESPACE_PATH  /DMQNS/
%EOS

%GNT
myglobalqueue  115.1  G
%EOS

%NAM
NA_GROUP 0
%EOS
```

Attempting to locate `myglobalqueue` without specifying `PSEL_TBL_BUS` in the namespace list results in a `PAMS__NOOBJECT` failure:

```
locate "myglobalqueue" with PSEL_TBL_GRP returns PAMS__NOOBJECT
```

However, attempting to locate `myglobalqueue` using `PSEL_TBL_BUS` in the namespace list results in success:

```
locate "myglobalqueue" with PSEL_TBL_BUS returns 115.1
```

Global names have the value in `DEFAULT_NAMESPACE_PATH` added as a prefix when they are cached. Therefore, the name will not be found in the local cache unless `PSEL_TBL_BUS` is either included in the namespace list or the fully prefixed name is specified:

```
locate "myglobalqueue" with PSEL_TBL_BUS returns 115.1
locate "myglobalqueue" with PSEL_TBL_GRP returns PAMS__NOOBJECT
locate "/DMQNS/myglobalqueue" with PSEL_TBL_GRP returns 115.1
locate "/DMQNS/myglobalqueue" with PSEL_TBL_PROC returns 115.1
locate "myglobalqueue" with PSEL_TBL_GRP,PSEL_TBL_BUS returns 115.1
```

Combining `PSEL_TBL_BUS` with `PSEL_TBL_GRP` results in the `DEFAULT_NAMESPACE_PATH` being automatically added as a prefix to the queue name before any table lookups begin.

If any of the following table names are included in the namespace list, then the `DEFAULT_NAMESPACE_PATH` is automatically added as a prefix to the queue name:

- `PSEL_TBL_DNS_CACHE`
- `PSEL_TBL_DNS_LOW`
- `PSEL_TBL_DNS_MED`
- `PSEL_TBL_DNS_HIGH`
- `PSEL_TBL_BUS`
- `PSEL_TBL_BUS_LOW`
- `PSEL_TBL_BUS_MED`
- `PSEL_TBL_BUS_HIGH`

If an application contains one of the above table names, but does not have a Naming Agent defined, then you must comment out `DEFAULT_NAMESPACE_PATH` in the `%PROFILE` section of the `DMQ$INIT.TXT` file to prevent the `DEFAULT_NAMESPACE_PATH` from being added as a prefix to the queue name.

If the `DMQ$INIT.TXT` file contains global queue name definitions, but does not have a Naming Agent defined, then those queue names are not resolved because global name lookups have been moved out of the process's context and into the Naming Agent. (This is different behavior than in BEA MessageQ for OpenVMS version 3.2B.)

The `DMQNS_DEFAULTPATH` logical name is supported and is defined in `DMQ$BOOT.COM`. Using the `DMQNS_DEFAULTPATH` logical name allows you to remove the `DEFAULT_NAMESPACE_PATH` definition from the `DMQ$INIT.TXT` file, yet still define the path information needed by the Naming Agent Server. The rules for defining this logical are the same rules documented for defining the `DEFAULT_NAMESPACE_PATH` in the `DMQ$INIT.TXT` file. The Naming Agent Server only uses this logical if a `DEFAULT_NAMESPACE_PATH` is not defined in the `DMQ$INIT.TXT` file. If the `DEFAULT_NAMESPACE_PATH` is defined, then the `DMQNS_DEFAULTPATH` logical is ignored.

Defining `DMQNS_DEFAULTPATH` allows name lookups for both local and global names to use the same `pams_locate_q` API call with a namespace list of `PSEL_TBL_PROC`, `PSEL_TBL_GROUP`, and `PSEL_TBL_BUS`. Assuming that the `DEFAULT_NAMESPACE_PATH` in `DMQ$INIT.TXT` is not defined, no path name is added as a prefix to the queue name at the API level. This provides flexibility for applications that require backwards compatibility with BEA MessageQ version 3.2B applications.

Below are some examples of global name caching using `pams_locate_q` with the following information defined in the group initialization file (`DMQ$INIT.TXT`) and the group boot file (`DMQ$BOOT.COM`).

`DMQ$INIT.TXT`

```
.
.
.
%PROFILE
*DEFAULT_NAMESPACE_PATH /DMQNS/ ! COMMENTED OUT
%EOS
.
.
.
%GNT
myglobalqueue          115.1    G
mylocalqueue           1.5      L
%EOS
.
.
.
%NAM
NA_GROUP 0
%EOS
```

`DMQ$BOOT.COM`

```
.
.
.
$!      -- Namespace root directory
$!
$       dmq_lnm DMQNS_DEVICE          dmq$disk:['root']
$!
$!      -- Namespace default path (based off of DMQNS_DEVICE)
$!
$       dmq_lnm DMQNS_DEFAULTPATH    "/DMQNS/"
.
.
.
```

Attempting to locate `myglobalqueue` without specifying `PSEL_TBL_BUS` in the namespace list will result in a `PAMS__NOOBJECT` failure:

`locate "myglobalqueue" with PSEL_TBL_GRP` returns `PAMS__NOOBJECT`

However, attempting to locate `myglobalqueue` with `PSEL_TBL_BUS` in the namespace list will result in a success:

```
locate "myglobalqueue" with PSEL_TBL_BUS returns 115.1
```

Global names are prefixed with the `DEFAULT_NAMESPACE_PATH` when they are cached. However, in this example the `DEFAULT_NAMESPACE_PATH` is not defined (or `NULL`). Therefore, the name will be cached without a namespace path prefix.

```
locate "myglobalqueue" with PSEL_TBL_BUS returns 115.1
locate "myglobalqueue" with PSEL_TBL_GRP returns 115.1
locate "/DMQNS/myglobalqueue" with PSEL_TBL_GRP returns PAMS__NOOBJECT
locate "/DMQNS/myglobalqueue" with PSEL_TBL_PROC returns PAMS__NOOBJECT
locate "myglobalqueue" with PSEL_TBL_GRP,PSEL_TBL_BUS returns 115.1
```

Attempting to locate `mylocalqueue` without specifying `PSEL_TBL_GRP` in the namespace list will result in a `PAMS__NOOBJECT` failure:

```
locate "mylocalqueue" with PSEL_TBL_BUS returns PAMS__NOOBJECT
```

However, attempting to locate `mylocalqueue` with `PSEL_TBL_GRP` in the namespace list will result in a success:

```
locate "myglobalqueue" with PSEL_TBL_GRP returns 1.5
```

Because the `DEFAULT_NAMESPACE_PATH` in this example is not defined (or `NULL`), there is no namespace path prefix appended on the queue name during lookup even though `PSEL_TBL_BUS` is in the namespace list.

```
locate "mylocalqueue" with PSEL_TBL_BUS returns PAMS__NOOBJECT
locate "mylocalqueue" with PSEL_TBL_GRP returns 1.5
locate "mylocalqueue" with PSEL_TBL_PROC returns 1.5
locate "mylocalqueue" with PSEL_TBL_GRP,PSEL_TBL_BUS returns 1.5
```

Defining Type and Class Codes

BEA MessageQ supports defining symbolic names for type and class codes within BEA MessageQ application programs and the BEA MessageQ Script facility. These codes are carried in the DMQ header file and are accessed using parameters of the API services.

The `DMQ$USER:DMQ$TYPCLS.TXT` file contains definitions of the following reserved type and class codes:

- MRS
- PAMS
- SBS_SERVER
- AVAIL_SERVER
- ETHERNET

It also contains definitions of:

- Generic UCB types and classes
- DEMO type codes for DEMO_IO_SERVER
- LU62 type codes for LU6.2 UCB
- Message types for LU6.2 services
- Message types for generic port server

You can add codes to the `DMQ$TYPCLS.TXT` file when you run `DMQ$CUSTOMIZE` (from the BEA MessageQ main menu) and enter `YES` in response to the following question:

```
Create language specific TYPE_CLASS include files [Y/N] (N)?
```

Listing 7-4 is an excerpt from the `DMQ$TYPCLS.TXT` file that shows the PAMS class and type codes section of the file.

Listing 7-4 Excerpt from DMQ\$TYPCLS.TXT

```
*****
*
*          PAMS   Class and Type codes (900-999)
*
*****
#MSG_CLAS_PAMS 29          * define PAMS class *

* define PAMS type codes *
msg_type_timer_expired      -900
msg_type_list_all_q_req     -960
msg_type_list_all_q_resp    -961
```

msg_type_enable_q_notify_req	-962
msg_type_enable_q_notify_resp	-963
msg_type_disable_q_notify_req	-964
msg_type_disable_q_notify_resp	-965
msg_type_q_update	-966
msg_type_locate_q_rep	-972
msg_type_linkmgt_req	-975
msg_type_linkmgt_resp	-976
msg_type_declare_sq	-980
msg_type_undeclare_sq	-981
msg_type_allocated_sq	-982
msg_type_msg_status	-983
msg_type_dmq_server_nak	-989
msg_type_enable_notify	-990
msg_type_disable_notify	-991
msg_type_process_dcl	-992
msg_type_process_exit	-993
msg_type_list_dcls	-994
msg_type_list_all_entries	-995
msg_type_list_all_connections	-996
msg_type_list_all_groups	-997
msg_type_link_lost	-998
msg_type_link_complete	-999

Table 7-2 describes types of statements used in the DMQ\$TYPCLS.TXT file.

Table 7-2 DMQ\$TYPCLS.TXT Statements

Statement Type	Description	
Comments	Begin with an asterisk (*) and can start anywhere on a line	
Blank lines	Transferred as is	
Symbol declarations	Type of Symbol	Prefix
	Message class	msg_clas_30000 through _32767
	Message type	msg_type_

Note: The only valid separators for this file are spaces and tabs.

8 Configuring the BEA MessageQ Client Library Server

This chapter describes how to configure and use the BEA MessageQ Client Library Server (CLS) on OpenVMS systems.

The following topics are covered in this chapter:

- BEA MessageQ Client Library Server Overview
- Client Library Server Installation and Transport Support
- Configuring the CLS section of DMQ\$INIT.TXT File
- Special Queue Configuration Issues
- Starting and Stopping CLS Manually
- CLS Event Logging and Tracing

BEA MessageQ Client Library Server Overview

The role of the BEA MessageQ Client Library Server (CLS) on OpenVMS systems is to provide applications access to message queuing capabilities without having BEA MessageQ servers running on the same system. Typically, these applications run on PCs, but BEA MessageQ also provides this economical access from any UNIX, OpenVMS, or Windows NT system. Client applications can communicate with other distributed application components anywhere in the network using a single connection with CLS to the message queuing bus architecture provided by BEA MessageQ.

The BEA MessageQ Client Library Server is implemented for all server platforms supported by BEA MessageQ. The CLS architecture is identical on all platforms. The CLS implementation is tailored for each operating system environment. See the *BEA MessageQ Client User's Guide* for your operating system for platform-specific information.

The BEA MessageQ Client Library Server for OpenVMS runs as a nonprivileged application program that uses BEA MessageQ for OpenVMS. CLS uses asynchronous message queuing operations and network system service routines to support multiple client connections using a single OpenVMS process. Support for multiple clients using a single process has the advantage of minimizing OpenVMS process resources compared to using one BEA MessageQ CLS process for each client.

All BEA MessageQ functions available to clients are supported by CLS. The return status codes from BEA MessageQ functions executed by the Client Library Server on OpenVMS are converted to the equivalent status values used by BEA MessageQ for UNIX or Windows NT. BEA MessageQ client applications on UNIX or Windows NT do not receive OpenVMS return status values.

Client Library Server Installation and Transport Support

The `VMSINSTAL` procedure for the BEA MessageQ for OpenVMS media kit installs the files needed to run the Client Library Server. The files specific to the Client Library Server are the following:

- `DMQ$LIB:CLS.OLB`
- `DMQ$EXE:DMQ$CLS.LNK`
- `DMQ$EXE:DMQ$CLS_START.COM`

The BEA MessageQ CLS supports DECnet and the following TCP/IP transports:

- DEC TCP/IP Services for OpenVMS
- Process Software Corporation (PSC) TCPware for OpenVMS
- Process Software Corporation (PSC) MultiNet TCP/IP for OpenVMS

During the BEA MessageQ for OpenVMS installation, the `LINKALL.COM` command procedure calls `DMQ$CLS.LNK` to link one or more `DMQ$CLS` Server image files. The `DMQ$CLS.LNK` command procedure creates an executable image file in the `DMQ$EXE` directory for all network transports available on your system. The CLS image file names are:

CLS Image File	Transport supported
<code>DMQ\$CLS_DECNET.EXE</code>	DECnet
<code>DMQ\$CLS_UCX.EXE</code>	DEC TCP/IP Services for OpenVMS
<code>DMQ\$CLS_TCP.EXE</code>	PSC TCPware for OpenVMS
<code>DMQ\$CLS_TGV.EXE</code>	PSC MultiNet TCP/IP for OpenVMS

When CLS is using the TCP/IP transport, the value of the BEA MessageQ logical name `DMQ$TCP_IP_LD` determines which transport-specific CLS image file runs.

Note: Refer to Chapter 3, “Configuring Cross-group Connections,” for details on how to define the `DMQ$TCP_IP_LD` logical name.

If you install a new TCP/IP transport product after installing BEA MessageQ, a new CLS image file must be linked to support the new TCP/IP transport. The BEA MessageQ development media kit can be used to create a new transport-specific CLS image file using the following command:

```
@DMQ$EXE : DMQ$CLS . LNK
```

The command creates a new CLS image file to connect clients using your transport product.

Configuring the CLS section of DMQ\$INIT.TXT File

Client Library Servers are started automatically by the BEA MessageQ COM Server using the `%CLS` section of the `DMQ$INIT.TXT` configuration file. Edit the Client Library Server configuration table as required. Listing 8-1 shows a sample CLS Configuration Table.

Listing 8-1 Sample Client Library Server Configuration Table

```
%CLS      **** Client Library Server Configuration Table ****
*
*
*  Endpoint      Transport      Maximum #      Security
*  -----      -
*           5000      TCP/IP          25      dmq$user:cl_5000.sf
*           5001      TCP/IP          25
*           6000      DECNET          32
```

Table 8-1 describes the parameter definitions for the CLS Configuration Table.

Table 8-1 CLS Configuration Table Parameter Definitions

Parameter	Value	Description
Endpoint	1024—65535	CLS TCP/IP port
Endpoint	1—99999	CLS DECnet object
Transport	DECNET or TCPIP	Transport type (DECnet or TCP/IP)
Maximum # of Clients	1—512	Maximum number of clients allowed for each CLS Server, OpenVMS only
Security File Path	<i>pathname</i>	The full pathname of the CLS Security File. If no security file is specified then CLS will not restrict access by remote clients.

Each entry in the table results in the creation of a multithreaded CLS process with the specified characteristics. For example, three Client Library Servers are started using the configuration table shown in Listing 8-1.

The first CLS process uses the TCP/IP transport and listens for connections on port 5000. The second CLS process also uses the TCP/IP transport, but listens for client connections using port 5001. Both servers support up to 25 client connections. The third CLS process uses the DECnet transport and accepts connections on the DECnet object name, DMQCLS_6000.

Each client application that connects to a CLS process uses BEA MessageQ message queuing resources on the OpenVMS system. Be sure to configure the BEA MessageQ buffer pools to support the messaging requirements of all BEA MessageQ applications running locally on the OpenVMS system, as well as the client applications connected to the message queuing bus by CLS.

CLS Endpoints

The CLS endpoint identifies either the TCP/IP port number or the DECnet object name the server uses to listen for client connections. The same endpoint is used in the configuration of BEA MessageQ clients to locate the Client Library Server.

When the transport type is TCP/IP, the available port numbers are in the range from 1024 to 65535. Port numbers less than 1024 are reserved for privileged applications, such as TELNET and FTP. There is no restriction on the use of port numbers within the available range by CLS. However, you should select port numbers that do not conflict with port numbers used by BEA MessageQ TCP/IP Link Drivers or other TCP/IP-based applications on your system.

When the transport type is DECnet, the available endpoint values are in the range from 1 to 99999. The endpoint value is concatenated with the prefix `DMQCLS_` to create a unique DECnet object name (for example, `DMQCLS_5000`). If the value of `Maximum # Clients` is equal to 1, then the endpoint range is reduced to values between 1 and 9999. This is used to support CLS subprocess object names.

Setting Maximum Number of Clients for CLS

The `Maximum # Clients` parameter determines the number of active client connections supported by a single CLS process. When the maximum number of clients are attached to CLS, additional attach queue attempts fail with the return status of `PAMS__REJECTED`. The maximum number of clients parameter is ignored by non-OpenVMS groups.

The optimum value for the maximum number of clients depends on the application. The maximum number of clients for each CLS is a function of variables unique to each application environment. Consider the following application characteristics when selecting a value:

- CPU processor type and system memory available

Faster machines with more memory allow individual CLS processes to handle a larger number of simultaneous client connections, depending on system load.
- Amount of BEA MessageQ message queuing activity by client applications

CLS can support a higher number of client connections if each client sends and receives messages with low frequency.
- Number of client waiting for messages

Multithreaded Client Library Servers use AST routines to support multiple client connections that perform blocking dequeue functions. As the number of clients waiting for messages increases, the CLS process incurs a gradually higher processing cost to receive each message.

- Number of network connections for each OpenVMS process

Each CLS process supports many network connections, each connection uses additional system and network resources.

- Number of separate CLS processes

Client applications share process and system resources of the CLS process. Different applications or users in different organizations may prefer to use separate CLS processes for easier application management.

Set the value of the `Maximum # Clients` parameter to a value that supports a large number of clients, then monitor the performance of the CLS process and the client messaging response. Add additional CLS processes as needed to meet the application load and system environment. Very large client networks can expect to use multiple Client Library Servers.

Special queue configuration issues arise using multithreaded Client Library Servers. The configuration issues are described in “Special Queue Configuration Issues.”

Configuring One Client for Each CLS Server Option

When the value of the `Maximum # Clients` parameter is set to 1, CLS supports multiple clients using one OpenVMS subprocess for each client connection. This configuration is referred to as a Single-Client mode. When `Maximum # Clients` parameter is greater than 1, CLS runs in multithreaded mode. When this parameter is equal to 1, CLS runs in multiprocess mode.

Note: When running the CLS in Single-Client mode, a separate CLS process is created for each client connection. Make sure the the sub-process quota (`PRCLM`) is large enough to accomodate the expected number of clients.

The Single-Client mode is used for remote BEA MessageQ client applications requiring unrestricted connection to any available queue, just like a local application. The BEA MessageQ for OpenVMS multithreaded CLS process provides client applications attachment to temporary queues and to permanent queues that are specifically defined as unowned secondary. See “Special Queue Configuration Issues” for more information.

Note that the permanent queues used by a CLS Server in the Single-Client mode do not need to be configured differently. Each client connection is handled by a separate OpenVMS process that functions like any local BEA MessageQ application process. The special queue configuration issues arise when using multithreaded CLS servers.

In Single-Client mode, the CLS process listens on the endpoint as a well-known location for client requests to connect. CLS behaves as a listener, waiting for connections.

When client connection requests arrive, the listener creates a separate OpenVMS Server subprocess to handle each client connection. This is similar to how the Client Library Server functions on UNIX systems. The OpenVMS process names for the CLS Server process begin with `CLS_S_`, followed by the listener endpoint number and a server sequence number. For example, the process name may be `CLS_S_05000_001`.

To reduce the delays for OpenVMS process creation, the CLS listener creates an available server process before it accepts the next client connection request. When clients connect to CLS, they first connect to the listener, which returns a dynamic port number for an available server subprocess. Clients disconnect from the listener and reconnect to the available server subprocess ready and waiting for their connection. BEA MessageQ client applications complete a `pams_attach_q` operation faster using a multithreaded CLS, compared to using a multiprocess CLS in the Single-Client mode.

Restricting Remote Access to CLS

The CLS security file is a text file that contains a table of client entries. Each client entry contains a list of endpoints and queues which the client may use. CLS uses the security file to restrict access by remote clients to those endpoints and queues. BEA MessageQ groups can have their own separate security files, or can share one jointly.

A template security file is available at the following location:

```
DMQ$DISK: {DMQ$V50.USER.TEMPLATE}DMQCLSEC.TXT
```

When you create a group, a copy of the template security file is automatically placed in the group's directory. Edit the file to remove the sample entries and add entries for the client systems in your environment. Then associate the security file with the message queuing group by specifying the Security File Path in the `%CLS` section of the group initialization file.

When a CLS is started, it loads the security file specified in the %CLS section of the group initialization file. If no security file is specified, CLS will not restrict access by remote clients. Each CLS can have a separate security file, or a security file can be shared by multiple CLS processes.

Special Queue Configuration Issues

Special queue configuration issues arise when using multithreaded CLS processes. Multithreaded CLS processes support client connections to both permanent and temporary queues. However, permanent queues used by clients must be configured differently in order to be supported by a multithreaded CLS process.

Permanent queues are defined in the Queue Configuration Table of the `DMQ$INIT.TXT` file. (See Chapter 4, “Configuring Message Queues and Global Memory,” for more information on the Queue Configuration Table.)

Permanent queues used by clients must be configured as unowned secondary queues for a multithreaded CLS process to support multiple clients. Unowned secondary queues are permanent queues with a `Q Type` of `S`, and a `Q Owner` specified by a period (`.`). (Normally, secondary queues have a `Q Owner` that identifies a primary queue that the queue is tied to.)

Listing 8-2 shows a sample of the Queue Configuration Table with permanent unowned secondary queues defined for use by CLS clients.

Listing 8-2 Sample Client Queues for Queue Configuration Table

```
%QCT          ***** Queue Configuration Table *****
*
*          ---Pool Quota---  UCB  Q   Q  Confrm Perm Name  Chk
*  Queue Name      Num  Bytes  Msgs Ctrl  Send Type Own  Style Act  Scope ACL
*  -----
*
*  Other permanent queues as needed.
*
*  Client Queues
*
CLIENT_1          1   64000  100 All    .   S   .   II   Y   L   N
CLIENT_2          2   64000  100 All    .   S   .   EI   Y   L   N
CLIENT_3          3   64000  100 All    .   S   .   EO   Y   L   N
CLIENT_4          4   64000  100 Msg    .   S   .   II   Y   L   N
CLIENT_5          5   64000  100 Msg    .   S   .   EI   Y   L   N
CLIENT_6          6   64000  100 Msg    .   S   .   EO   Y   L   N
CLIENT_7          7   64000  100 Byte  .   S   .   II   Y   L   N
CLIENT_8          8   64000  100 Byte  .   S   .   EI   Y   L   N
CLIENT_9          9   64000  100 Byte  .   S   .   EO   Y   L   N
CLIENT_10        10   64000  100    .   .   S   .   .   Y   L   N
*
*
%EOS
```

The multithreaded CLS process handles all differences between using permanent primary queues and permanent secondary queues. Client applications do not have to be changed to use secondary queues.

The differences in queue types handled by CLS include the following:

- Client applications attach `BY_NAME` or `BY_NUMBER` to permanent primary queues using the queue type parameter `PSYM_ATTACH_PQ`. Multithreaded CLS maps the permanent queue name or number from a primary queue (as viewed by the client application) to a secondary queue in the Queue Configuration Table.
- Client applications can also explicitly attach to one or more permanent secondary queues (`PSYM_ATTACH_SQ`), each of which are configured as unowned secondary queues. Implicitly attaching to owned secondary queues by attaching to the owning primary queue is not supported. (That is, secondary queues cannot be configured with a `Q OWNER` as another secondary queue used by a client.)

- Client applications define selection filters based on the primary queue to which they are attached. Multithreaded CLS maps the select filter to the appropriate operation on the secondary queue. (For example, when a client application calls `pams_get_msg` using the predefined select filter, `PSEL_PQ`, multithreaded CLS converts `PSEL_PQ` to an appropriate select filter to dequeue messages from the secondary queue used by that client.)
- Timer messages generated by a call to `pams_set_timer` that are normally delivered to the primary queue are automatically queued on the secondary queue used by the client.

In general, client applications are not aware that permanent queues are configured as unowned secondary queues instead of primary queues. All message queuing operations, including Message Recovery Services, functions in the same way.

Errors Attaching to Undefined Queues

Using a multithreaded CLS, client applications attempting to attach to permanent queues that are not defined in the Queue Configuration Table receive a return status value of `PAMS__NOTSECONDARYQ`, (`-270`). Because the client specified a queue type parameter of `PSYM_ATTACH_PQ`, the error is inconsistent. This is due to the queue type mapping performed by multithreaded CLS. However, it indicates that the queue must be defined as an unowned secondary queue in the Queue Configuration Table.

Starting and Stopping CLS Manually

For convenience and ease of application management, CLS Servers are generally started from the CLS Configuration Table when a BEA MessageQ group starts. CLS Servers also are stopped when the BEA MessageQ group is shut down. There are circumstances when new CLS Servers need to be started without affecting the entire BEA MessageQ group.

Starting CLS from the Manager Utility

A CLS Server process can be started from the Manager Utility, using the Start Service command on the Remote Client Management menu (Listing 8-3 and Listing 8-4). The service must already be defined in the %CLS section of the group initialization file. If the service is not defined in the group initialization file, then modify the initialization file to add the new service and then run the loader again using the following command:

```
$RUN DMQ$EXE:DMQ$LOADER.EXE
```

Listing 8-3 Manager Utility Remote Client Management Menu

```
Bus:99 Group:40 MessageQ Manager Utility Thu Jan 8 15:28:37 1998

Remote ClientLib Controls - Choose the function type
SS - Start Service          KS - Stop Service
EX - Exit Remote ClientLib Controls

Enter option:
```

Listing 8-4 Starting CLS from Remote Client Management Menu

```
Bus:99 Group:40 MessageQ Manager Utility Thu Jan 8 15:33:29
1998

Endpoint      Transport
5001          TCPIP
6000          DECNET

Enter end point number: 5001

Successfully started CLS Endpoint:5001, Transport:TCPIP
<CR> to continue:
```

Starting CLS from DCL

Additional CLS Server processes can be started using the DCL command procedure `DMQ$CLS_START.COM`, while BEA MessageQ is already running. This procedure is available in `DMQ$EXE`. Although CLS processes can be stopped and started without impacting the BEA MessageQ group, stopping a CLS process impacts all clients that are currently connected to that server.

To execute the `DMQ$CLS_START.COM` command procedure, enter

```
$ @DMQ$EXE:DMQ$CLS_START
```

This command procedure takes the following parameters:

```
$ @DMQ$EXE:DMQ$CLS_START endpt transport max clients
```

where

- *endpt* = endpoint
- *transport* = TCPIP or DECNET
- *max clients* = the maximum number of clients; 1 sets the Single-Client mode

Examples of `CLS_START` parameters:

```
@DMQ$EXE:DMQ$CLS_START 5000 TCPIP 25  
@DMQ$EXE:DMQ$CLS_START 5000 DECNET 25  
@DMQ$EXE:DMQ$CLS_START 5000 TCPIP 1
```

`DMQ$CLS_START` uses the `DMQ$EXE:DMQ$START_SERVER.COM` command procedure to create a detached CLS Server process.

Stopping CLS from the Manager Utility

A CLS Server process can be stopped from the Manager Utility, using the Stop Service command from the Remote Client Management menu (Listing 8-5), or by using the Force Process service (Listing 8-6).

Listing 8-5 Stopping CLS from Remote Client Management Menu

```
Bus:99 Group:40 MessageQ Manager Utility Thu Jan 8 15:33:29
1998
```

```
Endpoint      Transport
5001          TCPIP
6000          DECNET
```

```
Enter end point number: 5001
Successfully stopped CLS Endpoint:5001, Transport:TCPIP
<CR> to continue:
```

Listing 8-6 Stopping CLS from Force Process Menu

```
Bus:99 Group:40 MessageQ Manager Utility Thu Jan 8 15:48:58
1998
```

```
FORCEX Process
```

Index	Process Name	State	PQ Name	#Queues	#Pending
1	DMQ_N_009900040		NA_SERVER	1	0
2	DMQ_M_009900040		MRS_SERVER	1	0
3	DMQ_S_009900040		SBS_SERVER	1	0
4	DMQ_C_009900040		COM_SERVER	1	0
5	DMQ_T_009900040		TCPIP_LD	1	0
6	DMQ_D_009900040		DECNET_LD	1	0
7	DMQ_L_009900040		EVENT_LOGGER	1	0
8	DMQ_J_009900040		JRN_SERVER	1	0
9	DMQ_CLS_T_5001		+DMQ_CLS_T_5001_203	1	0
10	DMQ_CLS_D_6000		+DMQ_CLS_D_6000_204	1	0
11	_FTA30:		+_FTA30:_205	1	0

```
[End]
```

```
Index of process to be stopped? [1 - 11]:
```

CLS Event Logging and Tracing

The Client Library Server event logging is integrated with the BEA MessageQ for OpenVMS Event Logger process. CLS Server initialization and client connect/disconnect events are logged to the Event Logger.

The Client Library Server also supports trace output that provides more detailed information about client activity. The CLS trace output displays information about the BEA MessageQ API calls executed by CLS on behalf of client applications.

The CLS event log and trace output can be directed to the common event log, to the console, or to separate files. See Chapter 10, “Using BEA MessageQ System Management Utilities,” for more information on using the BEA MessageQ Event Log facility.

9 BEA MessageQ Main Menu and Utilities

The BEA MessageQ main menu enables you to run various management and configuration functions as well as restart and shut down the COM Server.

The following topics are covered in this chapter:

- Using the BEA MessageQ Main Menu
- Running the IVP
- Starting and Shutting Down the COM Server
- Verifying Buffer Pool Configuration (DMQ\$LLS_VERIFY)
- Testing BEA MessageQ Communications with DMQ\$LOOP
- Testing BEA MessageQ Services with DMQ\$TEST
- Running the CUSTOMIZE Command Procedure
- Running the DMQ\$LOADER Utility
- Changing Bus and Group Numbers
- Shutting Down BEA MessageQ

Note: The DMQ\$MGR_UTILITY and DMQ\$MONITOR utilities are described in Chapter 10, “Using BEA MessageQ System Management Utilities.”

Using the BEA MessageQ Main Menu

The BEA MessageQ main menu provides access to many BEA MessageQ command procedures and utilities. Before you access the BEA MessageQ main menu, make sure that you run the `DMQ$SET_LNM_TABLE` command procedure to connect to the group's environment.

After you start a BEA MessageQ group, invoke the BEA MessageQ main menu by entering the following commands:

```
$ DMQ : ==@DMQ$EXE : DMQ$MENU
$ DMQ
```

BEA MessageQ displays the main menu. Table 9-1 describes each BEA MessageQ main menu option and indicates where to find more information about each option.

Table 9-1 BEA MessageQ Main Menu Options

Option	Name	Description
1	Run IVP	Executes the Installation Verification Procedure. See "Running the IVP" and the <i>BEA MessageQ Installation Guide for OpenVMS</i> .
2	Restart/Shutdown DMQ\$COM_SERVER	Starts up or shuts down a group's BEA MessageQ Servers. See "Starting and Shutting Down the COM Server" and Chapter 2, "Defining the Message Queuing Environment."
3	Run LLS_VERIFY	Displays information about queues containing pending messages. See "Verifying Buffer Pool Configuration (DMQ\$LLS_VERIFY)."
4	Run LOOP	Performs a loopback of messages to test communication. See "Testing BEA MessageQ Communications with DMQ\$LOOP."
5	Run MONITOR	Monitors and controls BEA MessageQ communications. See Chapter 10, "Using BEA MessageQ System Management Utilities."

6	Run TEST	Sends and receives test messages. See “Testing BEA MessageQ Services with DMQ\$TEST.”
7	Run MGR_UTILITY	Displays queue information, flush a queue, force a process to exit, and display the group name table, as well as recovery journal management, logging and tracing control. See Chapter 10, “Using BEA MessageQ System Management Utilities.”
8	Run PSSVFY	Verifies that a BEA MessageQ script is syntactically correct. See the <i>BEA MessageQ Programmer’s Guide</i> .
9	@CUSTOMIZE procedure	Changes the system configuration by editing the DMQ\$BOOT.COM, DMQ\$INIT.TXT, or DMQ\$TYPCLS.TXT file. See “Running the CUSTOMIZE Command Procedure.”
10	Run LOADER	Adds a new name string, a new group, or changes the address associated with a named object. See “Running the DMQ\$LOADER Utility.”
11	Change Bus and Group numbers	Changes BEA MessageQ bus and group number for your process. See “Changing Bus and Group Numbers.”

Running the IVP

Select Option 1 on the main menu to perform the Installation Verification Procedure (IVP). You may run IVP to verify whether the BEA MessageQ software is functioning properly on your system.

Starting and Shutting Down the COM Server

Select Option 2 on the main menu to restart or shut down the DMQ\$COM_SERVER process. The system displays the following prompts:

```
Startup  DMQ_C_001500001 [Y/N] (N):
Shutdown DMQ_C_001500001 [Y/N] (N):
```

The default is NO. To perform either a BEA MessageQ startup or shutdown, enter Y.

Note: Refer to “Shutting Down BEA MessageQ” for more information on the shutdown process.

Verifying Buffer Pool Configuration (DMQ\$LLS_VERIFY)

The DMQ\$LLS_VERIFY utility maps a message queuing group's buffer pool configuration and verifies the consistency of data structures for the global section. It validates the configuration specified in the Buffer Pool Configuration Table of the DMQ\$INIT.TXT file.

To use this utility, select Option 3 on the main menu.

Testing BEA MessageQ Communications with DMQ\$LOOP

The DMQ\$LOOP utility tests message queuing bus communications by performing a loopback of messages, for both local and cross-group communications. This is a quick test to see if, and at what rate, messages of a specified size can be exchanged.

Note: The DMQ\$LOOP utility can only be used with other local and cross-group OpenVMS groups. This utility is not supported for testing cross-group communications with groups on UNIX or Windows NT systems.

Note: When running DMQ\$LOOP, large messages can fail if the screen queue quota on the OpenVMS system is too small.

Running DMQ\$LOOP

When running this utility, you perform the following steps:

1. Select Option 4 from the BEA MessageQ main menu to run DMQ\$LOOP.
2. Select L from the menu to perform the local loopback test. The system prompts you for the following data entry:

```
Message length (range 8:128000)?  
Number of messages?
```

When you enter the data, a statistical data screen is displayed. Press Return to continue.

3. Select x from the menu to perform the cross-group loopback tests. The system prompts for the following data entry:

```
Enter Target Group: 1  
Enter RCVW timeout [D:100]:  
Message length (range 8:128000)?  
Number of messages?
```

When you enter the data, an X-Group loop test statistics screen is displayed.

4. Select v from the menu to suppress (or activate) data verification. Data verification is generally ON. When you first select this option, the Data Verification switch changes from ON to OFF. The next time, it will be toggled to ON again.

Note: Use the data verification option to control writing and checking of a data pattern within each message. With small messages, this overhead is insignificant, however, if the message size increases, the overhead may become a large percentage of the cost of sending a message.

Testing BEA MessageQ Services with DMQ\$TEST

The `DMQ$TEST` utility tests BEA MessageQ API routines. It allows you to define, send, and receive messages interactively while being attached to a permanent or temporary queue. It also supports generating messages from a script.

To use this utility, choose Option 6 from the BEA MessageQ main menu. At the prompt, enter configuration information to set up the test.

Running the CUSTOMIZE Command Procedure

Select Option 9 from the BEA MessageQ main menu to access the `@CUSTOMIZE` command procedure. The `@CUSTOMIZE` option allows you to access and edit the `DMQ$BOOT.COM` and `DMQ$INIT.TXT` files to customize your BEA MessageQ configuration. The previous versions of the files should not be purged until the new configuration files have been tested. Note that the default action is to purge the files.

Customization Options

The `CUSTOMIZE` options are:

- Edit the `DMQ$BOOT.COM` startup file [Y/N]
Refer to “About the `DMQ$BOOT` Command Procedure” as well as Chapter 2, “Defining the Message Queuing Environment,” and Chapter 4, “Configuring Message Queues and Global Memory,” for more information on using the `DMQ$BOOT.COM` file.
- Edit the `DMQ$SET_SERVER_LOGICALS.COM` file [Y/N]

Enter `Y` and press Return to change information in the server logical command procedure. Refer to Chapter 12, “Managing a BEA MessageQ Environment,” for more information.

- Edit the `DMQ$SET_SERVER_QUOTA .COM` file [Y/N]

Enter `Y` and press Return to change information in the server quota command procedure.

- Edit the `DMQ$INIT.TXT` file [Y/N]

Refer to Appendix A, “Sample `DMQ$INIT.TXT` File,” for a sample of a complete `DMQ$INIT.TXT` file.

- Create language-specific `PROCESS, GROUP` include files [Y/N]

Enter `Y` to create new include files for `DMQ$INIT.TXT`. The following messages are displayed:

```
... Translating DMQ$INIT.TXT to language include files.
```

```
Translating for ADA
Translating for BASIC
Translating for BLISS
Translating for C
Translating for COBOL
Translating for FORTRAN
Translating for MACRO
Translating for PASCAL
Translating for PLI
Inserting files into library
```

- Edit `DMQ$TYPCLS.TXT` for `TYPE` and `CLASS` [Y/N]

Enter `Y` to add new classes and types. Refer to Appendix B, “Sample `DMQ$TYPCLS.TXT` File,” for the listing of all message types and class codes.

- Create language-specific `TYPE_CLASS` include files [Y/N]

Enter `Y` if you have edited the `DMQ$TYPCLS.TXT` file. The system displays the following message:

```
Translating DMQ$TYPCLS.TXT message class and type file into
ADA, BASIC, BLISS, C, COBOL, FORTRAN, MACRO, PASCAL and
PL/I....
```

```
Inserting files into library...
Rebuild PAMScript symbols [Y/N]
Enter Y to customize the BEA MessageQ Script Facility to your
```

present configuration.

The system displays the following message:

```
*This step takes from 1-30 minutes depending on system load, *  
*processor type, and number of symbols in the files. *
```

Submit to batch [S], Run at terminal [T]... [S,T] (T):

- Execute the customization procedure SYCUSTOMIZE.COM [Y/N] (N):

Enter Y to complete BEA MessageQ customization. Then press Return to return to the main menu.

SYCUSTOMIZE.COM can be modified to archive these changed modules:

```
DMQ$BOOT.COM  
DMQ$INIT.TXT  
DMQ$TYPCLS.TXT
```

9-OCT-1999 14:20:24

<CR> to continue:

About the DMQ\$BOOT Command Procedure

For each BEA MessageQ message queuing group, you can modify the DMQ\$BOOT command procedure either by using option 10 (@CUSTOMIZE) or editing the DMQ\$STARTUP procedure.

Use the logicals defined in DMQ\$BOOT to do the following:

- Configure a group to share configuration files with another group.
- Redirect the location of MRS files by defining the logical DMQ\$MRS to point to a different directory.
- Redirect output of BEA MessageQ console messages to a specific destination.
- Define the TCP/IP product that is used by the Link drivers and the Client Library Server.

Refer to Chapter 2, “Defining the Message Queuing Environment,” for more information on the DMQ\$BOOT.COM procedure.

About the DMQ\$INIT.TXT File

The `DMQ$INIT.TXT` file contains nine sections that you must customize for each group. Each section begins with a percent sign (%) following the name or an abbreviation for the name of the section. Refer to Chapter 2, “Defining the Message Queuing Environment,” for instructions on how to customize the group initialization file.

About the DMQ\$TYPCLS.TXT File

BEA MessageQ software customization includes defining the symbolic names for type and class codes within BEA MessageQ application programs and the BEA MessageQ script facility. The `DMQ$TYPCLS.TXT` file contains symbolic definitions of type and class codes.

See Chapter 7, “Creating Global Names,” for more information on how to edit this file.

Running the DMQ\$LOADER Utility

You can dynamically reload the `DMQ$INIT.TXT` file at any time, without stopping BEA MessageQ, by running the program `DMQ$EXE:DMQ$LOADER.EXE`. Refer to “Changing Parameters in the Running Group” in Chapter 2, “Defining the Message Queuing Environment,” for information on changing parameters in a running group.

Starting the LOADER Utility

Select Option 10 on the main menu to start the LOADER utility.

Restrictions on DMQ\$LOADER

Note the following restrictions when using `DMQ$LOADER`:

- `DMQ$LOADER` supports only add and modify operations, but not deletes. `DMQ$LOADER` has no delete capability, but allows a symbol to be set to a null value (0).
- Changes to the `%BUFFER`, `%MRS`, or `%SBS` Server initialization sections require that the group be restarted before changes take effect.

Changing Bus and Group Numbers

Select Option 11 to change BEA MessageQ buses and groups on your process.

The following is an example of the prompt the system may display (for Bus 15):

```
Enter Bus ID to change to [0015] [? for list]:
```

```
Enter Group ID to change to [? for list]:
```

Enter a question mark (?) to display the list of existing LNM tables for the node.

Shutting Down BEA MessageQ

Use Option 2 to select the COM Server shutdown command. The system displays:

```
Startup  DMQ_C_001500001 [Y/N] (N):
```

```
Shutdown DMQ_C_001500001 [Y/N] (N):
```

At the Shutdown `DMQ_` prompt, enter Y.

Do not use the DCL command `STOP` to stop the COM Server.

When BEA MessageQ shuts down, it performs the following tasks:

- The exit handler forces all attached processes to exit.
- The exit handler ensures that global sections are deleted.
- On exiting, the COM Server sets a bit to cause other processes to exit when the global sections are accessed.

- The SYS\$OUTPUT device for the attached process displays this message:

```
%PAMS-F-PAMSDOWN, This MessageQ
group is down -
      COM Server has exited
```


10 Using BEA MessageQ System Management Utilities

This chapter describes two BEA MessageQ utility programs that can be used to perform system management tasks and to monitor BEA MessageQ system status.

- Using the Monitor Utility (`DMQ$MONITOR`)
- Using the System Management Utility (`DMQ$MGR_UTILITY`)

The `DMQ$MGR_UTILITY` and `DMQ$MONITOR` utilities require that the BEA MessageQ environment variables be defined and accessible to the user. Use the `DMQ$STARTUP.COM` command procedure to define these variables.

The `DMQ$MONITOR` utility must be attached to a running group, while only some options of the `DMQ$MGR_UTILITY` utility require a connection to a running group. You can run these utilities independently or access them from the BEA MessageQ main menu.

Major System Management Tasks

Table 10-1 and Table 10-2 list major BEA MessageQ system management tasks and options that you can select using the `DMQ$MONITOR` and `DMQ$MGR_UTILITY` utilities. Refer to corresponding sections for detailed information on listed features.

Table 10-1 lists DMQ\$MONITOR options.

Table 10-1 DMQ\$MONITOR System Management

Tasks	Option to Use
To display queue information:	
Display counts of messages sent, messages received and messages pending for attached queues	D
Display the same information for all configured queues	D
Continuously monitor queue states and counters	D
Display information about message queue quotas	Q
Display queue-specific information	P
Reset dynamic counts of number of messages received and messages sent since the last reset operation	R
To display group and network connection information:	
Display cross-group network message routing information.	RT
Re-establish lost cross-group connections	C
Display network transport communications links for a group	LS
Display detailed [inbound or outbound] link information	LD
To display group information:	
Display detailed group information	GD
Change to alternate group	A
Display system status and resource allocation statistics	T
Display failover availability for message groups	LT
To shut down a group:	
Shut down the group	K

Table 10-2 lists all major BEA MessageQ system management tasks you can perform using `DMQ$MGR_UTILITY`.

Table 10-2 DMQ\$MGR_UTILITY System Management Tasks

Tasks	Option to Use
To control and display queues and processes:	
Display processes connected to a group, attached queues, and programs that run	DS
Display quotas and counters of an attached queue, along with PID, process name, and program image name of the process attached to the queue	DQ
Cause a process attached to BEA MessageQ to exit by invoking the <code>SYS\$FORCEX</code> system service	FR
Flush all messages pending to an unattached queue	QC
Start or stop queues	QC
Display names in the group name space	DG
Start or stop Client Library Services	CM
To control event and error logging and tracing:	
Enable or disable tracing for a process attached to BEA MessageQ	RO
Redirect event, error logging, and/or trace output from a specific process to any of several targets, including a common event log, and the system console device	RO
Close the common event log file and open a new one	LC
To control message recovery journals:	
Open and close DQF and SAF streams	JC
Close a DLJ or PCJ stream and open a new one	JC
Enable or disable all recovery streams (DQF, SQF, DLJ, and PCJ) in a given group	JC

Tasks	Option to Use
Show what recovery streams have messages in them and what files are associated with the streams	JC
Obtain a formatted dump of the contents of an inactive (closed) recovery stream	JC
Redirect a recovery stream associated with one queue to another target queue, either replacing the target stream, or merging with it	JC
To use the Link Management Subsystem to control cross-group network links:	
Shut down a network link to another BEA MessageQ group	LM
Start a network link to a BEA MessageQ group	LM
To use the Naming Agent Subsystem to manage names and namespaces	
Display information about names and namespaces	NA
Manage the association of names and addresses	NA
Remove names from the namespace	NA

Using the Monitor Utility (DMQ\$MONITOR)

To run the Monitor utility, select Option 5 from the BEA MessageQ main menu.

The Monitor utility is used to monitor and control the BEA MessageQ communications system. Because this utility uses messages to request information from the COM Server, you can use it to monitor more than one message queuing group. The Monitor utility screens display the current system status and configuration information.

Displaying Queue Counters

This option provides the display of permanent queue names (from DMQ\$INIT.TXT) and temporary queue names.

Select Option D from the Monitor utility menu to display the current number of messages that are sent, received, or pending in a message queue.

The system prompts you to choose:

```
Display all, declared or MRQ [A,D,M] [A]:
```

where:

[All]	Use to select all queues.
[Declared]	Use to select attached queues.
[MRQ]	Use to select multireader queues (that share Read access).

Displaying Queue Quotas

Use this option when monitoring memory usage to determine whether an application is reading its messages as fast as they are being sent. You can assign greater system resources to applications that are falling behind in their message reading.

Select Option Q from the Monitor utility menu to display queue quotas. The system prompts you to choose:

```
Display all, declared or MRQ [A,D,M] [A]:
```

where:

[All]	Use to select all queues.
[Declared]	Use to select attached queues.
[MRQ]	Use to select multireader queues (that share Read access).

Displaying COM Server Status

Select Option T from the Monitor utility menu to display statistics of the COM Server process. The COM Server Specific Information screen displays messages sent to and received by the COM Server process, errors logged to the COM Server, the amount of memory available in the buffer pool, and the number of queue slots remaining. Use this option to review system status and resource allocation to avoid such problems as buffer pool exhaustion and queue failures.

Press Return to return to the Monitor utility menu options.

The COM Server Specific Information screen displays the following information:

- **MCS created on** shows the date and time when the group was created.
- **Counters reset on** shows the last time the queues and group counters were reset by either a Monitor Reset command or an automatic reset when the counters exceeded a counter-full threshold.
- **DMQ ID** shows the platform type and DMQ version..
- **COM Server Msgs** shows standard queue counts for the COM Server's queue. It also shows the number of times the COM Server became active to scan the buffer pool.
- **Error Section** shows the number of:
 - Entries in the error log
 - Queuing failures encountered which typically exhaust the buffer pool
 - Process cleanups that the COM Server needed to perform on behalf of a failed process to assure that global queuing resources were released
- **Buffer Pool** shows the current status of the BEA MessageQ buffer pool. It shows the total number of buffers, number of unallocated buffers, and the threshold of available count. The Low Buffer and user program access protection warnings are displayed including an asterisk (*) in the warning level column if the buffer pool is Low.
- **Queue Slots** shows the current number of allocated temporary queues and the total of remaining slots.

- **DmQ cross group communication** shows if communications between BEA MessageQ groups has been enabled. If `DISABLED` is displayed, no message exchange can occur between BEA MessageQ groups.

Displaying Queue-Specific Status

Use this option to display the queue number, the process ID, and the number of messages sent, received, and pending for the selected queue.

Select Option P from the Monitor utility menu to display the status of a particular queue. The system prompts:

```
Enter queue # for request (default = COM Servers queue number):
```

Resetting COM Server Counters

Select Option R from the Monitor utility menu to reset the COM Server counters. By resetting the counters to zero, you can create a starting point from which to begin monitoring COM Server statistics.

Displaying the Routing Table

Select Option RT from the Monitor utility menu to display routing table information.

The routing table is part of the `DMQ$INIT.TXT` file and is used to configure cross-group communications between groups that do not have a direct network connection.

Shutting Down COM Server Process

Select Option K from the Monitor utility menu to shut down the COM Server process. Use this option to provide an orderly shutdown of cross-group communications. To use this option, you must have the `ACCEPT_KILL_CMD` set to `YES` in the group initialization file and have either `VMS OPER` privilege or the Manager utility rights

identifier `DMQ$OPERATOR`. See “Defining Access Control on Queues” in Chapter 13, “BEA MessageQ Security,” for more information on the `DMQ$OPERATOR` rights identifier.

Caution: Do not use the `DCL STOP` command to shut down the COM server. Refer to Chapter 9, “BEA MessageQ Main Menu and Utilities,” on other ways to stop the COM Server.

Displaying Link Summary Information

Select Option `LS` from the Monitor utility menu to display link summary information. The system prompts you for the group number from which to start. This screen displays the status of DECnet and TCP/IP network communications links for each group.

Displaying Link Detail Information

Select Option `LD` from the Monitor utility menu to display detailed link information. The system asks you to choose:

```
Inbound or Outbound? (I/O) [I]
```

Make your selection; the default is Inbound (I).

If the network link uses a TCP/IP transport, the fields in the lower right corner of the screen display would show:

```
Transport type = TCP/IP
Host name      = rabbit
INET address   =
Port number    = 4573
```

If the network link has a transport type unknown to the Monitor utility, the fields in the lower right corner of the screen display would show:

```
Transport type = *Unknown* (n)
Host name      = rabbit
Host address   =
Connect address = 4573
```

Displaying Link Connect Table

Select Option LT from the Monitor utility menu to display the link connect table. This screen displays the failover search list for groups as defined in the `DMQ$INIT.TXT` file.

Displaying Group Detail Information

Select Option GD from the Monitor utility menu to display group detail information.

Resetting Cross-Group Connections

Select Option C from the Monitor utility menu to re-establish cross-group connections.

Use this option if network communications links between the groups were lost and you want to re-establish all cross-group connections.

Displaying Remote Groups

Select Option A from the Monitor utility menu to monitor the status of a remote group.

After you set the alternate group by using this option, all options will display the information for the selected group. To display information for the local group, select this option and enter the local group number.

Note: This option is valid only for message server systems; it is not available for client implementations.

Using the System Management Utility (DMQ\$MGR_UTILITY)

The DMQ\$MGR_UTILITY program provides a collection of menu-driven services that manage and control several BEA MessageQ subsystems.

To run this utility, select Option 7 from the BEA MessageQ main menu.

Displaying Queue Summary Information

Select Option DS from the Manager utility menu to display summary level information about queues.

Displaying Queue Detail Information

Select Option DQ from the Manager utility menu to display detailed information about a particular queue. The utility prompts for the queue number.

Flushing Queues

Select Option QC from the Manager utility menu, then select Option FQ from the Queue Controls menu to remove any pending queue message entries from a message queue.

Use this option to selectively clear queues of pending messages, which can remain after an abnormal program termination. By removing unwanted messages from the unattached queues, you clear space in the buffer pool for other messages.

When you select this option, the system prompts for the queue number to flush.

Note: The queue to be flushed can not be attached when you initiate the flush request.

Stopping Queues

Select Option QC from the Manager utility menu, then select Option SPQ from the Queue Controls menu to stop all message traffic to the specified message queue. After a Stop Queue request has been executed, no more messages are allowed to be enqueued into the stopped queue until the queue is restarted.

When you select this option, the system prompts for the queue number to be stopped and inquires whether to stop the queue fast (immediately) or gracefully.

```
Enter queue number to stop:  
Stop fast? (Y/N) [N]:
```

When a queue is stopped gracefully, the queue will no longer accept new messages; however, any pending messages that are in the queue are allowed to be read until the queue is empty. After the queue is empty it is placed in a stopped state.

When a queue is stopped fast, the queue will not accept any new messages and will not allow pending messages to be read from the queue.

Starting Queues

Select Option QC from the Manager utility menu, then select Option STQ from the Queue Controls menu to start stopped message traffic to the specified message queue. When you select this option, the system prompts for the queue number to be started.

Forcing a Process to Exit

Select Option FR from the Manager utility menu to force a process to exit.

Use this utility instead of the DCL command `STOP` to allow process exit handlers to control a clean process shutdown.

Redirecting Status and Trace Output

Error and status logging in BEA MessageQ for OpenVMS can be enabled or disabled by using:

- logical names that are read when a process calls the `pams_attach_q` service
- the RO option, dynamically, from the Manger utility.

Enabling Tracing Prior to Starting a Program

To enable tracing prior to starting a user or sender program, set either one or both of the following logicals:

- `DMQ$DEBUG = ALL`
Turns on the BEA MessageQ call tracing.
- `DMQ$SERVER_TRACE = YES`
Turns on the service-specific tracing for a BEA MessageQ Server process.

In either case, trace output is sent to the targets which are specified in the logical name `DMQ$TRACE_OUTPUT`. The following targets are valid:

Target	Description
<code>EVL_LOG</code>	Common event log for the group
<code>CONSOLE</code>	Console TTY device, either <code>OPA0:</code> or <code>DMQ\$ERROUT:</code>
<code>SYSOUT</code>	<code>SYS\$OUTPUT:</code> for the process
<code>SYSERR</code>	<code>SYS\$ERROR:</code> for the process
<code>USER_LOG</code>	A file in <code>SYS\$LOGIN:</code> or the file specified by <code>DMQ\$USER_LOG_NAME</code>
<code>MEM_LOG</code>	In-memory ring buffer
<code>NONE</code>	No trace output sent

The default setting for DMQ\$TRACE_OUTPUT is SYS\$OUTPUT. You can override the default settings by changing the process logical name values prior to running a program. This is done for BEA MessageQ Servers by modifying the DCL command procedure DMQ\$USER:DMQ\$SET_SERVER_LOGICALS.COM. See Listing 10-1.

Listing 10-1 Default MRS section in DMQ\$USER:DMQ\$SET_SERVER_LOGICALS.COM

```
$ MRS:
$   trace_output      = "SYSOUT"
$   group_output     = "EVL_LOG,CONSOLE"
$   process_output   = "SYSOUT"
$   debug            = ""
$   headers          = ""
$   server_trace     = ""
$   user_log         = ""
```

Enabling Tracing When a Program is Running

Use the RO option of the DMQ\$MGR_UTILITY to dynamically enable the tracing feature and redirect tracing information without stopping and restarting the application.

Enter the primary queue number of the process you want to modify.

Using this screen, you can also select debug switch settings (DMQ\$DEBUG), which allow you to set output streams for the selected debugging information, as follows:

ERROR	Display errors only
TRACE	Display call tracing
ALL	Display complete (error and call tracing) data
NONE	Disable tracing

Note: This option is also used for event logging using DMQ\$PROCESS_OUTPUT and DMQ\$GROUP_OUTPUT settings. Refer to Chapter 12, “Managing a BEA MessageQ Environment,” for more information on separating error logging streams to isolate system problems.

Event Logging

The central error logging facility manages the Event Logger process (EVL_LOG). This facility directs error and debugging trace information to output streams: EVL_LOG, CONSOLE, SYSOUT, SYSERR, USER_LOG, MEM_LOG, or NONE. These streams are assigned by logical names, with defaults provided by BEA MessageQ. They can be dynamically redirected from the Redirect Status/Trace Switch option.

Directing Error and Status Messages Prior to Starting a Program

Two separate classes of events can be logged by a process connected to BEA MessageQ; a process event and a group event. Process events apply only to a particular process while group events are the events that impact the entire message queuing group.

Process events are logged to the targets which are specified in the logical name DMQ\$PROCESS_OUTPUT. This logical can be set to any or all of the same targets that DMQ\$TRACE_OUTPUT can be set to: EVL_LOG, CONSOLE, SYSOUT, SYSERR, USER_LOG, MEM_LOG, or NONE. The default for DMQ\$PROCESS_OUTPUT is SYS\$OUTPUT.

Group events are logged to the targets which are specified in DMQ\$GROUP_OUTPUT. Any, or all of the five targets may be chosen. The default for DMQ\$GROUP_OUTPUT is EVL_LOG,CONSOLE.

Redirecting Error and Status Logging When a Program is Running

Event logging can be dynamically redirected with the RO option of DMQ\$MGR_UTILITY. Refer to “Redirecting Status and Trace Output” for more information. Note that the RO option prompts you to set values for the DMQ\$PROCESS_OUTPUT and DMQ\$GROUP_OUTPUT logical names.

Using Event Log Control

The event log file is a common log managed by the Event Logger. It serves as a single repository for logging all BEA MessageQ events. All entries are time-stamped.

Storing Event Data

Any trace data or events which are directed to the `EVL_LOG` target are sent by the mailbox to the `DMQ$EVENT_LOGGER` process. This server is designed to service the mailbox as quickly as possible, while efficiently handling logging to the `DMQ$LOG:DMQ$EVL_bbbb_gggg.LOG` file. This mechanism means that high logging event rates can be handled with less impact on the timing of events than if the events were logged synchronously in the process context. This feature therefore can be useful in tracing the application or BEA MessageQ problems that are time-related.

Switching Event Log Files

Select Option LC from the Manager utility menu to run the Event Log Control. Use this option to close and move the `EVL_LOG` file, as well as to start a new log

The LC event log control option is used to switch logging from one version of the log to the next version.

Closing a log file and opening a new log file allows the previous version to be copied to tape or disk, and then be deleted. This option can be used by sites with 24x7 operations that can not afford to shut down a running group to manage large event log files. A large event log can be generated in unusual situations where large number of network or queuing errors are repeated due to some system failure.

Note: Refer to Chapter 12, “Managing a BEA MessageQ Environment,” for more information on system troubleshooting.

Displaying Group Name Table

Select Option DG from the Manager utility menu to display all entries contained in the group name table for the current message queuing group.

The Time column shows the last time each entry was updated. The updates take effect when the `DMQ$LOADER` utility runs.

Journal Controls

Select Option JC from the Manager utility menu to select the Journal Controls menu. This menu allows you to control all journal management functions.

BEA MessageQ provides the following journal files:

- Destination Queue File (DQF)
- Store and Forward File (SAF)
- Dead Letter Journal (DLJ)
- Post Confirmation Journal (PCJ)

Journal Control selections allow you to open and close the message recovery journals and provide formatted dumps of their contents. All journal files (DQF, SAF, PCJ, and DLJ) can be manipulated with the `DMQ$MGR_UTILITY` program. These recoverable and auxiliary journal files have the same disk format; each journal is a sequence of files, not a single file).

Note: Refer to Chapter 5, “Configuring Message Recovery,” for detailed information on the BEA MessageQ message recovery system and journal files.

Managing DQF and SAF Journals

When you select either the DQF or SAF option, the system displays a menu allowing you to open, close, delete, merge, dump, redirect, and obtain a directory of SAF or DQF files.

SAF or DQF files are dynamically created and deleted as needed by the recovery system; they are not preallocated fixed-size files. Both DQF and SAF files may be closed and the stream of messages in the file redirected to other targets. This can be in response to a message to the MRS Server from a suitably privileged user program, or from the entry in the Journal Control menu.

The DQF and SAF controls include the following options:

Note: Refer to Chapter 14, “Managing Failover,” for more information on Merge and Redirect options.

[Open]	Opens the DQF or SAF journal for a group and queue.
--------	---

[Close]	Closes the DQF or SAF journal. Note: When a DQF or SAF journal is closed, inbound traffic for the journal is rejected and the UMA forced. The journal must be reopened to accept the traffic.
[Delete]	Deletes the DQF or SAF journal associated with the queue.
[Merge]	Transfers and merges the contents of a DQF or SAF journal to another DQF or SAF journal. Used during a failover.
[Dump]	Performs a formatted dump of a DQF or SAF journal file.
[Directory]	Displays a directory of DQF or SAF journal files.
[Redirect]	Redirects recoverable data stream and opens a DQF or SAF journal at another group. Used during a failover.

Managing DLJ and PCJ Journals

When you select either the DLJ or PCJ option, the system displays a menu that allows you to dump, delete, redirect, or display a directory of auxiliary journals.

The DLJ and PCJ journals are served by a separate server process, the JRN Server. This process maintains a history of all PCJ and DLJ filename changes in a disk file. This history is displayed by the DI (directory) and DU (dump) commands of DMQ\$MGR_UTILITY to make it easy to find an old journal.

The PCJ and DLJ controls include the following options:

[Directory]	Displays a directory of DQF or SAF journal files.
[Dump]	Performs a formatted dump of a DQF or SAF journal file.
[Exit]	Exits a journal.
[Delete]	Deletes the DQF or SAF journal associated with the queue. Deleting an earlier journal removes its entry from the journal history file. Deleting all the existing journals clears the history file.
[Switch to]	Closes an old journal and opens a new one.

Managing the Naming Agent and the Namespace

Naming provides the ability to separate application processing from configuration details by allowing applications to refer to queues by name. The Naming Agent accesses and manages the BEA MessageQ bus-wide namespace. A namespace is the repository where names and their associated queue addresses are stored.

Select Option NA from the Manager utility menu to display the Naming Agent menu, as shown in Listing 10-2. This menu allows you to control all Naming Agent functions. Each option first asks for a group number associated with the Naming Agent server. Enter 0 for the local group (the default value), or enter a number from 1 to 32000 that corresponds to a message queuing group. The message queuing group that runs the Naming Agent must be identified in the %NAM section of the group initialization file.

Listing 10-2 Naming Agent Menu

```
Bus:1 Group:1 MessageQ Manager Utility Mon Nov 22 17:21:25 1999
      Naming Agent Management - Choose a function type
      LA - List All Entries          PA - Show Path
      SA - Set Address              ZA - Zero Address
      GE - Get Entry                DE - Delete Entry
      EX - Exit Naming Agent Management
      Enter function :
```

The following sections describe the individual naming agent management functions. For the purposes of these examples, the %GNT section of the DMQ\$INIT.TXT file is specified as shown in Listing 10-3.

Listing 10-3 Group Name Table Section of DMQ\$INIT.TXT File

```
%GNT ***** Group Name Table Section *****
*
```

* Queue Name	Group.Queue	Scope
*-----	-----	-----
GLOBAL_QUEUE1	1.234	G
* %EOS		

Displaying Information About Names and the Namespace

Select Option LA from the Naming Agent Management menu to list all the name entries in the name space that match the search criteria. The system prompts for a group number associated with the Naming Agent. The system prompts you for search criteria. The default is an asterisk (*), which is a wildcard character and results in a list of all entries in the namespace. You can also enter a specific name or portion of a name, including a wildcard. Listing 10-4 shows the List All Entries function.

Listing 10-4 List All Entries Function

```
Bus:44  Group:42  MessageQ Manager Utility  Tue Dec 14 15:30:56 1999

      Naming Agent Management - Choose a function type

      LA - List All Entries          PA - Show Path

      SA - Set Address              ZA - Zero Address

      GE - Get Entry                DE - Delete Entry

      EX - Exit Naming Agent Management

      Enter function : LA
Enter NA Group (def=0, local):
Enter List Search (def=*)      :

      GLOBAL_QUEUE1

<CR> to continue:
```

Select Option GE from the Naming Agent Management menu to display information about a specific name. The system prompts you for an entry name. Listing 10-5 shows the Get Entry function.

Listing 10-5 Get Entry Function

```
Bus:44  Group:42 MessageQ Manager Utility  Tue Dec 14 15:30:56 1999

      Naming Agent Management - Choose a function type

      LA - List All Entries          PA - Show Path

      SA - Set Address              ZA - Zero Address

      GE - Get Entry                DE - Delete Entry

      EX - Exit Naming Agent Management

      Enter function : GE
Enter NA Group (def=0, local):
Enter Entry name (def=back) : GLOBAL_QUEUE1

Name : /DMQ$DISK/DMQ$V50/DMQNS/GLOBAL_QUEUE1
Address : 1.234
Time last modified : Tue Dec 14 15:27:55 1999

<CR> to continue:
```

Select Option PA from the Naming Agent Management menu to display the namespace path. Listing 10-6 shows the Show Path function.

Listing 10-6 Show Path Function

```
Bus:44  Group:42 MessageQ Manager Utility  Tue Dec 14 15:30:56 1999

      Naming Agent Management - Choose a function type

      LA - List All Entries          PA - Show Path

      SA - Set Address              ZA - Zero Address

      GE - Get Entry                DE - Delete Entry

      EX - Exit Naming Agent Management

      Enter function : PA
```

```
Path is : /DMQNS/
```

```
<CR> to continue:
```

Managing Address and Name Associations

Select Option ZA from the Naming Agent Management menu to clear an address associated with a name. The system prompts you for an entry name for which the address is to be cleared. Listing 10-7 shows the Zero Address function.

Listing 10-7 Zero Address Function

```
Bus:44  Group:42 MessageQ Manager Utility  Tue Dec 14 15:30:56 1999

      Naming Agent Management - Choose a function type

      LA - List All Entries          PA - Show Path
      SA - Set Address              ZA - Zero Address
      GE - Get Entry                DE - Delete Entry
      EX - Exit Naming Agent Management

      Enter function : ZA
Enter NA Group (def=0, local):
Enter Entry name (def=back) : GLOBAL_QUEUE1

<CR> to continue:
```

Select Option SA from the Naming Agent Management menu to associate an address (*group.queue*) with a name. The system prompts you for an entry name and an address to be associated with the name. Listing 10-8 shows the Set Address function.

Listing 10-8 Set Address Function

```
Bus:44  Group:42 MessageQ Manager Utility  Tue Dec 14 15:30:56 1999

      Naming Agent Management - Choose a function type
```

```
LA - List All Entries      PA - Show Path
SA - Set Address           ZA - Zero Address
GE - Get Entry            DE - Delete Entry
EX - Exit Naming Agent Management

Enter function : SA
Enter NA Group (def=0, local):
Enter Entry name (def=back) : GLOBAL_QUEUE1
Enter Address (group.queue) : 543.210

<CR> to continue:
```

Removing Names from the Namespace

Select Option DE from the Naming Agent Management menu to delete a specified name from the namespace. The system prompts for the name to be deleted from the namespace.

Listing 10-9 Delete Entry Function

```
Bus:44 Group:42 MessageQ Manager Utility Tue Dec 14 15:30:56 1999

Naming Agent Management - Choose a function type

LA - List All Entries      PA - Show Path
SA - Set Address           ZA - Zero Address
GE - Get Entry            DE - Delete Entry
EX - Exit Naming Agent Management

Enter function : DE
Enter NA Group (def=0, local):
Enter Entry name (def=back) : GLOBAL_QUEUE1

<CR> to continue:
```

11 Sizing and Tuning the BEA MessageQ Environment

BEA MessageQ software uses the capabilities of the underlying operating system and networking software to perform message queuing. For this reason, you may need to change the system configuration to run BEA MessageQ and BEA MessageQ applications efficiently. You may also need to tune your system on a regular basis to accommodate the growth and changes in your BEA MessageQ environment.

The primary BEA MessageQ resource to configure is memory—both global and local virtual memory. Other system resources, such as disk files, network links, and system parameters may need to be adjusted for large queuing networks to function properly. This chapter describes how to set OpenVMS and BEA MessageQ configuration parameters and quotas for proper BEA MessageQ operation in the following topics:

- Sizing and Tuning Processes
- Allocating Virtual Memory for BEA MessageQ Servers
- Global Memory
- Tuning TCP/IP for BEA MessageQ
- Configuring BEA MessageQ for Large Messages

Sizing and Tuning Processes

This section describes how to configure:

- Virtual Memory
- Global Memory
- I/O Channels
- Files
- Network Resources
- Other System Resources and Quotas
- Modifying `DMQ$SET_SERVER_QUOTAS.COM`

Virtual Memory

Virtual memory is allocated in OpenVMS by setting the page file quota allotted to a OpenVMS process. This quota must be less than the size of the system `PAGEFILE` and the system parameter `VIRTUALPAGECNT`.

The `VIRTUALPAGECNT` parameter limits the total number of pages that a single process can map to itself. The amount of virtual memory that is resident at any time is determined by the working set size, which is usually set much smaller than the page file quota.

The page file quota is passed as a parameter to the `RUN` command used to detach a BEA MessageQ Server process. The quota for each server is specified in the `DMQ$USER:DMQ$SET_SERVER_QUOTAS.COM` command procedure. Methods for determining the amount of memory needed for specific servers is described in “Modeling Memory Usage for Each BEA MessageQ Server.”

Global Memory

BEA MessageQ for OpenVMS uses four global sections to store messages. The MCS global section stores message headers and the other three global sections, called the LLS for small, medium, and large sections, store messages.

The amount of global memory allotted to store messages is a fixed resource. Available global memory is determined by the number and size of the buffer pool specified in the %BUFFER section of the DMQ\$INIT.TXT file.

BEA MessageQ also uses global sections for storing group information. These global sections are sized based on entries in the DMQ\$INIT.TXT file. BEA MessageQ run-time libraries also require global memory.

Adding a new BEA MessageQ message queuing group may require changes to OpenVMS system parameters to allow the addition of global sections. The OpenVMS system parameters affected by this type of change are GBLPAGES, GBLPAGFIL and GBLSECTIONS.

BEA MessageQ has quotas to limit the amount of global memory used by a particular queue. Queue quotas limit the number of messages queued and the total number of bytes that can be queued.

I/O Channels

The OpenVMS CHANNELCNT parameter limits the number of I/O channels allotted to a single process. Channels are used by BEA MessageQ for network links and for message recovery journal files.

The COM Server allocates two channels for each cross-group link it manages. The TCP/IP and DECnet Link Driver processes allocate four channels per link. The MRS Server requires a channel for each open area file, which is a fixed size file used to store recoverable messages.

Files

The BEA MessageQ message recovery system dynamically creates and deletes files as needed. The number of files that MRS can use is limited by the `FILLM` quota which is passed to the `RUN` command that starts the MRS Server. The total number of files used by the MRS Server is also limited by the OpenVMS `CHANNELCNT` parameter.

Network Resources

The primary system resource needed for cross-group messaging is buffer space associated with each cross group link. Buffers are stored in the local memory of the COM Server or Link Driver processes. Refer to “Allocating Virtual Memory for BEA MessageQ Servers” for more information.

To maintain simultaneous DECnet connections to more than 32 message queuing groups, you must adjust the NCP parameter `MAXLINKS`. Similarly, for TCP/IP networks, the maximum number of sockets must be increased if an insufficient number are available. Refer to “Computing the Number of TCP/IP Sockets” for information on how to adjust this setting. In addition, large networks with many network links require an increase in system nonpaged pool to provide more memory for device drivers.

Other System Resources and Quotas

The COM Server and Link Driver processes use the `$QIO` interface to post AST service requests for network I/O. Each outstanding AST counts against the process quota called `ASTLM`. Timers are also associated with network requests and count against the `TQELM` quota.

The MRS Server uses the `$QIO` interface to post AST service requests for both read and write operations to a recoverable disk file. Timers are also used and associated with each unconfirmed message.

Modifying DMQ\$SET_SERVER_QUOTAS.COM

Each server's process quotas and limits, process name, and server-specific output files are defined in the `DMQ$SET_SERVER_QUOTAS.COM` file located in the `DMQ$USER:` directory. The `DMQ$SET_SERVER_QUOTAS.COM` command procedure can be edited to modify the system quotas assigned to any BEA MessageQ Server process.

Listing 11-1 shows the quota information for the COM Server process.

Listing 11-1 COM Server Quotas

```

$ COM:
$ proc_name == "DMQ_C_''comp_id'"
$ full_name == "COM Server"
$ img_file  == "DMQ$EXE:DMQ$COM_SERVER.EXE"
$ log_file  == "'dmq_log'DMQ$COM_SERVER_''full_id'.LOG"
$ prio      ==      6 !process software priority
$ biolm     ==      500 !buffered I/O limit (counts outstanding operations)
$ diolm     ==      500 !direct I/O limit (counts outstanding operations)
$ buflm     == 500000 !buffered I/O byte limit
$ tqelm     ==      500 !timer queue elements
$ enqlm     ==      500 !enq / deq locks
$ fillm     ==      500 !open files
$ astlm     ==      500 !Pending ASTs
$ subprocs  ==      16 !child sub processes
$ pgflquo   == 30000 !virtual memory
$ wsextent  ==    8192 !limit borrowing beyond wsquo
$ wsquo     ==    1024 !basic working set limit in pages
$ wsdef     ==      750 !starting working set size
$ goto FINISHED

```

Allocating Virtual Memory for BEA MessageQ Servers

Proper allocation of virtual memory resources is critical to successful and efficient processing in the BEA MessageQ environment. This section describes how to determine appropriate virtual memory allocation and shows how to model memory usage for each BEA MessageQ Server.

BEA MessageQ Servers are designed to continue operating if available virtual memory is exhausted. An operation requiring more memory than is available will fail; however, the server will continue to operate. If the server cannot deliver nonrecoverable messages, they are discarded. If the server cannot deliver a recoverable message, BEA MessageQ executes the Undeliverable Message Action.

To determine the appropriate amount of virtual memory for your BEA MessageQ configuration:

- Develop a rough memory usage model and validate it.
- Simulate the system under load.
- Measure the actual system under load.

Modeling Virtual Memory Needs

A rough model of memory usage requirements can be constructed by adding the memory requirements of all components managed by a server. The objects a server must track are determined by the data flow and timing of the system. This section provides a sample calculation for the MRS Server.

The amount of virtual memory used by a server can be obtained using the `DCL SHOW PROCESS /CONTINUOUS` command. The maximum amount of memory used by a server is also written to a server's output log file when the group is shut down. Size the rough model by configuring a minimum server, measuring the memory it requires, and adding the memory requirements of its queues, groups, and links.

Performing Testing

After you model the system and estimate the amount of virtual memory required, you can build a network of simple message senders and receivers that send at rates that you expect the real application to encounter, or you can test the production applications under expected system load.

During testing, virtual memory exhaustion is logged as an error message in the group's event log. If errors are encountered, increase the virtual memory allocated to the server and rerun the tests until the error no longer occurs.

Modeling Memory Usage for Each BEA MessageQ Server

To see how memory varies with the addition of a new group:

- Measure the virtual memory usage of a running group.
- Add a new group to the `DMQ$INIT.TXT` file.
- Connect the Group (run the `DMQ$LOADER` program after changing `DMQ$INIT.TXT`).
- Measure the change to virtual memory usage with the new group.

In the BEA MessageQ OpenVMS environment, the objects are tracked by each server. Table 11-1 shows the objects tracked by the COM Server.

Table 11-1 COM Server

Object	Memory Size Determined By
Code, fixed overhead	Measuring the minimum configuration
Groups	Varies with the number of groups on the bus
Network buffers	Varies with the number of connected groups. Sized per group in the <code>%XGROUP</code> section of the <code>DMQ\$INIT.TXT</code> file.
Queues	Local memory data structures used in attaching/detaching queues

11 Sizing and Tuning the BEA MessageQ Environment

Table 11-2 shows the objects tracked by link drivers.

Table 11-2 Link Drivers

Object	Memory Size Determined By
Code, fixed overhead	Measuring the minimum configuration
Groups	Varies with the number of groups on the bus
Network buffers	Varies with the number of active links. Sized per group in the %XGROUP section of the DMQ\$INIT.TXT file

The COM Server and link drivers share a common memory allocation mechanism to handle network buffers. The following formula roughly calculates this value:

```
pool_size_in_pages =  
sum of XGROUP pool buffer sizes in Kbytes from the %XGROUP section  
of DMQ$INIT.TXT. Multiply by 2 to convert to pages.
```

```
network buffers =  
48 guard pages + pool_size_in_pages + 2 * large buffer size in pages
```

Table 11-3 shows the objects tracked by the MRS Server.

Table 11-3 MRS Server

Object	Memory Size Determined By
Code, fixed overhead	Measuring the minimum configuration
Groups	Varies with the number of groups on the bus
Queues	Varies with the number of recoverable queues both local and remote
Messages	Varies with the number of unconfirmed messages
Internal buffers	Varies with largest message size

I/O data structures	<p>Measuring the size of the I/O data structures. The size varies with the size of the largest message. Assigned per each target queue with a recoverable message. To measure this:</p> <ul style="list-style-type: none"> ■ Measure the MRS Server virtual memory before sending a any recoverable message ■ Send a recoverable message ■ Measure the difference
---------------------	--

Table 11-4 shows the objects tracked by the Journal Server.

Table 11-4 Journal Server

Object	Memory Size Determined By
Code, fixed overhead	Measuring the minimum configuration
Internal buffers	Varies with largest message size
I/O data structures	One per stream. The Journal Server manages two streams, the PCJ stream and the DLJ stream. The sizing will be less than that required for the MRS Server since the Journal Server does not read the files. The size of the I/O data structures varies with the size of the largest message.

Note: The Journal Server uses the same I/O mechanism as the MRS Server, but does not allocate read ahead buffers since it does not read.

Table 11-5 shows the objects tracked by the SBS Server.

Table 11-5 SBS Server

Object	Memory Size Determined By
Code, fixed overhead	Measuring the minimum configuration
Groups	Varies with the number of groups maximum 512 groups
Avail registrations	Varies with the number of avail/unavail registrations
Broadcast registrations	Varies with the number of broadcast registrations

11 Sizing and Tuning the BEA MessageQ Environment

Multicast targets	Index that allow quick access from a MOT to a broadcast registration
Ethernet buffers	Varies with the number of MOTs assigned to a multicast address

Example Memory Allocation Model for the MRS Server

Table 11-6 shows an example memory allocation model for the MRS Server using parameter values taken from a specific release of BEA MessageQ. This model serves only as an example, along with an example configuration for a hypothetical network. Actual values are release dependent; therefore, it is important to check the product release notes.

Table 11-6 Memory Allocation Model for MRS Server

Component	Values
Page size	512 bytes
Code, RTL's, core messaging	< 10,000 pages (measured)
I/O buffer size	$(\text{large_message_size} + \text{page_size}) / \text{page_size}$
Cross group information	1/4 page per group
Per queue information	1 page per queue
Per message overhead	1/2 page per unconfirmed message
Overhead ($\text{large_msg_size} / \text{open area}$)	<ul style="list-style-type: none">■ 32,000 or greater/85 pages■ 16,000/49 pages■ 8,000/29 pages■ 4,000/20 pages

BEA MessageQ for OpenVMS Version uses a strategy in which I/O is addressable at a per-block level and achieves speed by use of asynchronous \$QIO calls. The overhead per each open area is determined by the number of RMS data structures and buffers

needed to handle the largest logical operation, and by the number of read ahead operations allowed. Large messages have the single greatest effect on the virtual memory requirements of the MRS Server.

The following example shows how to obtain the memory requirement. For example, if the MRS requires from 1 to 5 areas open for each stream, assume the following:

- a `large_msg_size` of 32000, with a topology of 10 groups, 5 queues per group, full recoverable traffic between each queue.
- for any group, there are 50 recoverable streams, 40 SAFs and 10 DQFs.
- the maximum unconfirmed message depth is application dependent. For SAF files, it will be limited to the number of senders times the number of remote queues or $5 \text{ senders} * 10 \text{ groups} * 5 \text{ queues} = 250$.
- $\text{remote message overhead} = 250 \text{ messages} * 1/2 \text{ page per msg} = 125 \text{ pages}$
The maximum unconfirmed message depth is determined by queue quotas. Assume 100 messages per queue (a common default setting). Then, $5 \text{ DQFs} * 100 \text{ unconfirmed messages} / \text{DQF}$ produces the following:

Local message overhead	250 pages.
Fixed overhead	10000 pages
IO_buffer_size	64 pages 1 large message
XGROUP connections	3 pages 10 groups / 4
Queues	50 pages $50 * 1 \text{ page per queue}$
Areas	21250 pages $50 * 5 \text{ areas} * 85 \text{ pages}$
Messages	375 pages see above discussion
Total pages required	31742 pages

For this application, sizing the MRS virtual memory at 32,000 pages should be sufficient. The default provided is 30,000; therefore, the `DMQ$SET_SERVER_QUOTAS.COM` file must be modified.

Global Memory

All message queuing groups on a node use the shared images `DMQ$EXECRTL` and `DMQ$ENTRYRTL`. Each individual group creates nine global sections. They are:

- A single `DMQ$MCS_C_bbbb_ggggg` Message Control section. This section is fixed in size. The size is dependent upon the release of BEA MessageQ.
- A `DMQ$LLS_L_bbbb_ggggg` section which increases depending upon the number and size of large messages.
- A `DMQ$LLS_M_bbbb_ggggg` section which increases depending upon the number and size of medium messages.
- A `DMQ$LLS_S_bbbb_ggggg` section which increases depending upon the number and size of small messages.
- A `DMQ$GNT_C_bbbb_ggggg` section which increases with the setting of the Profile parameter `NAME_TABLE_SIZE`.
- A `DMQ$MRQ_C_bbbb_ggggg` section which increases with the number of MRQs in the `%QCT` of `DMQ$INIT.TXT`.
- A `DMQ$GRP_C_bbbb_ggggg` section which increases with the number of groups.

Global Sections

The `GBLSECTIONS` parameter limits the total number of global sections that can be used at one time. The first message queuing group that you start up on your system uses eighteen global sections. Each additional group creates nine global sections for every COM Server that is running.

Global Pages

The `GBLPAGES` parameter defines the total number of pages that global sections use in the virtual memory.

Global Page File

The `GBLPAGFIL` parameter defines the total number of pages that global sections can take up in the page file. All dynamic BEA MessageQ global sections are paged to the page file.

Tuning TCP/IP for BEA MessageQ

If the network chosen for cross-group connections is DEC TCP/IP (formerly called UCX), then TCP/IP may need to be tuned to support the increased load of network traffic caused by running BEA MessageQ. In general, OpenVMS nonpaged pool and the number of TCP/IP sockets may need to be increased.

Approximating the Nonpaged Pool Needs

DEC TCP/IP requires a minimum of 500,000 bytes of additional nonpaged pool, and an additional 2,000 bytes for each socket. (For more information, see the *DEC TCP/IP Services for OpenVMS Installation and Configuration Guide*.) To determine the amount of additional nonpaged pool that will be needed for BEA MessageQ, you must allow for large buffers and group connections.

Use the following formula to determine the approximate worst case needs of BEA MessageQ:

$$npp = ((6 * (\text{Large DMQ buffer} + 323)) + 2,000 * (\text{grp_connections} + 1))$$

To adjust the amount of nonaged pool in the system configuration, modify the following parameters in the MODPARAMS.DAT file:

```
ADD_NPAGEDYN = npp
ADD_NPAGEVIR = npp
```

Computing the Number of TCP/IP Sockets

To determine the number of additional sockets required, multiply the number of group connections by 2. Add this number to the total number of available sockets on the system. To view the current number of sockets, use the following command:

```
$ UCX SHOW COMMUNICATIONS
```

To change the value of the socket setting, use the following command:

```
$ UCX SET COMMUNICATIONS/DEVICE_SOCKET=n
```

Improper configuration of TCP/IP sockets may result in a `EXCEEDQUOTA` error logged by the TCP/IP Link Driver.

Configuring BEA MessageQ for Large Messages

There are several parameters that must be changed in the BEA MessageQ configuration files to provide support for large messages up to 4MB. In some cases, OpenVMS system parameters must also be changed.

Note: Although BEA MessageQ supports message sizes up to 4MB, SBS direct Ethernet broadcasting (optimized Ethernet mode) does not support messages larger than 32K. SBS Datagram Broadcasting supports large messages up to 4MB.

Maximum Message Size

In the `%PROFILE` section of the `DMQ$INIT` file, set the `GROUP_MAX_MESSAGE_SIZE` to the largest message size which will be used by the group. It must be less than or equal to the value 4,194,304.

Message Pool Configuration

In the `%BUFFER` section of the `DMQ$INIT` file, the size and quantity of the message pool is important for large message processing. When a message is requested that is larger than the `LARGE` buffer size, which is normally 64,000, then the required number of buffers are chained together to hold the message. For a message size of 4,194,304, a total of 66 `LARGE` buffers of 64,000 bytes are required to hold their message. This configuration of the buffer pool directly affects the size of the global section created by COM server upon boot. The size of this buffer pool can exhaust a normal OpenVMS configuration, typically resulting in the following output:

Configuring BEA MessageQ for Large Messages

```
DmQ T 28:21.1 Time Stamp - 2-NOV-1999 10:28:21.18
DmQ T 28:21.1 ----- COM Server Starting -----
DmQ I 28:21.2 COM Server (V5.0-80) starting at 2-NOV-1999 10:28:21
DmQ F 28:21.4 Fatal error while creating Large LLS Pool
DmQ F 28:21.4 %SYSTEM-F-EXGBLPAGFIL, exceeded global page file limit
%SYSTEM-F-EXGBLPAGFIL, exceeded global page file limit
```

```
$help/message EXGBLPAGFIL
```

```
EXGBLPAGFIL, exceeded global page file limit
```

```
Facility:      SYSTEM, System Services
```

```
Explanation:   The attempt to allocate a global section with page file
                backing store failed because the systemwide limit on these
                pages is exceeded. No part of the section is allocated.
```

```
User Action:   Delete some similar sections or ask the system manager
                to increase the SYSGEN parameter GBLPAGFIL. Then, try the
                operation again.
```

In this case, edit the OpenVMS system parameters to increase the SYSGEN parameter GBLPAGFIL.

In some cases, the COM server cannot start and the system displays information similar to the following example:

```
DmQ T 41:09.8 Time Stamp - 2-NOV-1999 10:41:09.84
DmQ T 41:09.8 ----- COM Server Starting -----
DmQ I 41:09.8 COM Server (V5.0-80) starting at 2-NOV-1999 10:41:09
DmQ F 41:10.1 Fatal error while creating Large LLS Pool
DmQ F 41:10.1 %SYSTEM-F-GPTFULL, global page table is full
%SYSTEM-F-GPTFULL, global page table is full
```

```
$help/message GPTFULL
```

```
GPTFULL, global page table is full
```

```
Facility:      SYSTEM, System Services
```

```
Explanation:   Not enough space is available in system memory to maintain
                information about global sections. This message indicates a
                system error resulting from insufficient allocated space in
                the global page table.
```

```
User Action:   Notify your system operator or system manager to increase the
                SYSGEN parameter GBLPAGES.
```

11 Sizing and Tuning the BEA MessageQ Environment

In this case, edit the OpenVMS system parameters to increase the `SYSGEN` parameter `GBLPAGES`.

Buffer Pool Parameter

In the `%XGROUP` section of the `DMQ$INIT` file, the `Buf Pool` value must be greater than that largest message that will be sent to this group. The value is only used for sending messages to other groups. The purpose of the pool is to gain concurrence over network links; however, as CPU speed increased, this buffering is significant only in specialized cases. Nevertheless, for large cross group message, such as 4MB, the `Buf Pool` value must be set to at least 4,200 (in thousands).

Queue Quota

The queue quota must either be large enough to receive a large message buffer, or it must be disabled. For example, when sending messages between groups on OpenVMS systems, you can use `DMQ$LOOP` to test sending large messages. Because `DMQ$LOOP` uses a temporary queue, change the temporary queue quota in the `DMQ$INIT` file to either be large enough to handle the increased buffer size, or set the method to `NONE`.

Global Section Size

In the `DMQ$USER` area, edit the `pgflquo` parameters in the `DMQ$SET_SERVER_QUOTAS.COM` file to allow the BEA MessageQ processes to handle the increased global section size. For the COM server and for link drivers, set the value to 90,000 or 100,000. If recoverable messages are being used, increase the `MRS pgflquo` value depending on the number of messages in the pipe. The value may need to be set to 200,000.

Timeouts

Timeouts must be increased. BEA MessageQ applications that expect message throughput in seconds or fractions of seconds must be changed to allow for the increased time required for large message transfer.

Message Recovery Services

If the average message size will be 4MB, adjust the following parameters:

- Increase `AREA_SIZE` to a minimum of 8000 blocks in the `DMQ$INIT` file.
- Increase `CHUNK_SIZE` to 32767 in `DMQ$SET_SERVER_LOGICALS.COM`.
- Either increase the byte quota on the receiving queue, or set the quota checking to `Msgs`, in the `DMQ$INIT` file. Quota checking must be enabled on a receiving MRS queue for the `CACHE_PERCENTAGE` receive quota to work correctly.

11 *Sizing and Tuning the BEA MessageQ Environment*

12 Managing a BEA MessageQ Environment

BEA MessageQ problems can affect an entire message queuing bus, one or more message queuing groups, or a single BEA MessageQ application. This chapter describes how to troubleshoot problems that affect the operation of your BEA MessageQ environment.

Processing problems in the BEA MessageQ environment are generally caused by:

- Inadequate or incorrect configuration
- A hardware or software error underlying any portion of the BEA MessageQ environment
- Application development problems where BEA MessageQ behavior was different than what the application expected

The information in this chapter will help you to determine whether a problem lies with the BEA MessageQ environment or is application-specific. It describes:

- BEA MessageQ error logging
- Tools for troubleshooting BEA MessageQ problems
- Troubleshooting procedures

For more information on problems encountered during application development, refer to the *BEA MessageQ Programmer's Guide*.

BEA MessageQ Error Logging

Before you begin troubleshooting BEA MessageQ problems, you need to understand how BEA MessageQ alerts you to error conditions. This section describes how BEA MessageQ logs informational and error messages, how BEA MessageQ Servers operate, and how to pinpoint the source of the problems you encounter.

BEA MessageQ Output

BEA MessageQ offers several mechanisms to provide BEA MessageQ system managers, maintainers, application developers, and users with information about the status of BEA MessageQ and BEA MessageQ applications.

BEA MessageQ Stream Output

BEA MessageQ outputs messages to inform users about the current status of processing and to record system events. BEA MessageQ can display informational, status, and error messages on a terminal screen, print them on the operator's console, or write them to a log file.

BEA MessageQ messages are designed to serve a number of purposes. Some messages assist in debugging an application. Some messages alert the user to an event, error, or potentially serious problem affecting the entire BEA MessageQ group. Output messages are grouped according to their output streams.

There are three BEA MessageQ output streams. Each stream can be referred to by a logical name as follows:

Stream	Logical Name	Used for
Trace	DMQ\$TRACE_OUTPUT	Debugging
Process	DMQ\$PROCESS_OUTPUT	Informing the application or user of events or errors of interest ONLY to that application or user.
Group	DMQ\$GROUP_OUTPUT	Informing BEA MessageQ Servers or other applications of events or errors.

Both the process and trace streams are of local interest only because they report information or events pertaining to a single application. The group stream output is of interest to the entire message queuing group because it reports information or events which can affect operation of the entire group. Each BEA MessageQ process, whether an internal server or user application process, uses these three output streams.

Stream Destinations

A stream can send output to up to five destinations. A destination can be a terminal screen, operator console, or a log file. The assignment of a stream's destination(s) can be made at either group or application startup.

Stream output can also be changed and redirected using the BEA MessageQ Manager utility (DMQ\$MGR_UTILITY) Redirect Output (RO) menu option. Redirecting output includes adding or removing a destination to a stream's output or creating a new output file.

The destinations for stream output are:

- `YSOOUT`—The standard output device (`YSO$OUTPUT`).
- `YSERR`—The standard error reporting device (`YSO$ERROR`).
- `USER_LOG`—A file opened by the user's process. This file takes either the name specified by the user in the `DMQ$USER_LOG_NAME` logical name or the default. `YSO$LOGIN:DMQ$UL_qlname_bbbb_ggggg.log` is the default (*qlname* is the name of the queue, *bbbb* is the bus id and *ggggg* is the group id).
- `EVL_LOG`—The common event log file `DMQ$LOG:DMQ$EVL_bbbb_ggggg.LOG`, where *bbbb* is the bus id and *ggggg* is the group id.
- `CONSOLE`—The console device (`OPA0`, `OPCOM` or an operator-enabled terminal).
- `MEM_LOG`—The in-memory trace buffer. The trace information is stored in memory and is only written to disk when the group or process is stopped.
- `NONE`—No stream output is sent.

Stream output is directed to a destination based on the value set for the `DMQ$TRACE_OUTPUT`, `DMQ$PROCESS_OUTPUT`, or `DMQ$GROUP_OUTPUT` logical names. If you want to direct output to multiple destinations, assign a string containing the destinations (separated by commas) to the logical name.

12 Managing a BEA MessageQ Environment

For example, you would assign `YSOUT`, `USER_LOG`, `CONSOLE` to the `DMQ$GROUP_OUTPUT` logical name if you want the group stream to output to your terminal, the user log file, and the system console. The system will take default values, if the stream logical names are not defined.

Stream Switches

Some types of stream data require switches to be set to enable the output. A switch is set if a particular string has been assigned to the corresponding logical name. Output which falls into this category (and their corresponding logical names) are:

Output Type	Switch Logical	Values	Value Interpretation
Debugging Trace	DMQ\$DEBUG	ERROR	Display Error messages to DMQ\$TRACE_OUTPUT
		TRACE	Display Trace messages to DMQ\$TRACE_OUTPUT
		ALL	Display both Error and Trace messages to DMQ\$TRACE_OUTPUT
Display of Message Header Data	DMQ\$HEADER	YES	Display BEA MessageQ message headers.
Server Tracing (for DMQ Server processes only)	DMQ\$SERVER_TRACE	YES	Display server trace messages.

The Event Logger

Each BEA MessageQ group has an Event Logger Server which logs events, errors, and informational messages to a central file called `DMQ$LOG:DMQ$EVL_bbbb_ggggg.log`. These messages include announcements of server process startup, the failure of any portion of a BEA MessageQ group, or any log message sent to the event logger by a BEA MessageQ application. To uncover the source of a BEA MessageQ problem, always begin by reading the event log.

Console Output

Stream output sent to the console is normally sent to the central operator console. This output can be redirected by setting the value of the `DMQ$ERRROUT` logical name.

`DMQ$ERRROUT` specifies the destination for warning messages from the BEA MessageQ Servers. Messages written to `DMQ$ERRROUT` are also logged to the appropriate log file.

In-Memory Logging

The `MEM_LOG` value indicates that in-memory tracing is enabled. Tracing information is only written to disk when the group or process is stopped.

The size of the in memory trace buffer is a compile-time constant and is set to receive 20,000 lines of 120 characters in length. The buffer is configured as a ring, therefore only the last 20,000 events are captured.

When the trace routine is first called, it initializes the ring buffer and establishes an exit handler, which is called when the program terminates. When the program is terminated, the exit handler writes out the contents of the trace buffer to the `DMQ$LOG:MEMORY_TRACE.LOG` file.

BEA MessageQ server tracing is enabled either by the command procedure `DMQ$USER:DMQ$SET_SERVER_LOGICALS.COM` or the BEA MessageQ Manager Utility (`DMQ$MGR_UTILITY`). To enable in-memory tracing using the Manager Utility, set the `trace`, `group`, and `process output` fields to `MEM_LOG` and enable one or more tracing options.

It is recommended that the page file quota be increased for the server process that will be using the in-memory tracing.

Enabling Tracing

You can use the `DMQ$DEBUG` and `DMQ$SERVER_TRACE` logical names to enable tracing prior to starting a user or sender program. For more information, see “Enabling Tracing Prior to Starting a Program.”

You can use the `RO` option of the `DMQ$MGR_UTILITY` to dynamically enable tracing and redirect tracing information without stopping and restarting the application. For more information, see “Enabling Tracing When a Program is Running.”

BEA MessageQ Servers

Internal processing of BEA MessageQ messages is performed by several group server processes. Each server runs as an OpenVMS detached process. As separate processes, each server has its own process quotas and limits and its own stream output.

The main overseer of each BEA MessageQ group is the COM Server. Upon group startup, the COM Server will create other server processes. Whether the COM Server creates a other server process is based upon each server's entry in the `DMQ$INIT.TXT` file.

BEA MessageQ Server Output Files

Each server process has a log file where its output is written. The log files are contained in the directory specified by the `DMQ$LOG` logical name. The content of these files can be viewed using the OpenVMS `TYPE` command. Server log files are very useful for monitoring resources used by the servers and traceback information in the event of a server crash.

Table 12-1 lists the servers and their corresponding output files. In the file names, *bbbb* is the bus ID, *ggggg* is the group ID, and *eeee* is the endpoint. In the `CLS` file name, *x* is either `S` for single-client mode, `D` for DECnet multi-client mode, or `T` for TCP/IP multi-client mode:

Table 12-1 Servers and Output Files

Server Process	File
Com Server	<code>DMQ\$COM_SERVER_bbbb_ggggg.LOG</code>
SBS Server	<code>DMQ\$SBS_SERVER_bbbb_ggggg.LOG</code>
MRS Server	<code>DMQ\$MRS_SERVER_bbbb_ggggg.LOG</code>
Event Logger	<code>DMQ\$EVENT_bbbb_ggggg.LOG</code>
Journal Server	<code>DMQ\$JRN_SERVER_bbbb_ggggg.LOG</code>
Naming Agent Server	<code>DMQ\$NA_SERVER_bbbb_ggggg.LOG</code>
QTransfer Server	<code>DMQ\$QTRANSFER_bbbb_ggggg.LOG</code>

TCPIP Link Driver	DMQ\$TCPIP_LD_bbbb_ggggg.LOG
DECnet Link Driver	DMQ\$DECNET_LD_bbbb_ggggg.LOG
Client Library Server	DMQ\$CLS_x_eeee_bbbb_ggggg.LOG

BEA MessageQ Server Logging/Debugging

BEA MessageQ offers the same stream output mechanisms for server processes as are available for user processes. The logical names used to direct output streams are contained in the `DMQ$SET_SERVER_LOGICALS.COM` procedure located in the `DMQ$USER` directory.

To isolate and diagnose a server problem, you can modify a server process stream output to direct specific server output such as server debugger tracing. Listing 12-1 is a sample of the COM Server portion of the `DMQ$SET_SERVER_LOGICALS.COM` file.

Listing 12-1 COM Server Logical Name Settings

```

$ COM:
$   trace_output      = "SYSOUT"
$   group_output     = "EVL_LOG,CONSOLE"
$   process_output   = "SYSOUT"
$   debug            = " "
$   headers          = " "
$   server_trace     = " "
$   user_log         = " "
$   goto SET_LOGICALS

```

The symbol definitions in this file equate to the `DMQ$*` stream logical names for that particular server. For example, the `trace_output` symbol above equates to the `DMQ$TRACE_OUTPUT` logical name for the COM Server. When the `DMQ$SERVER_TRACE` logical name is set to `YES`, server-specific debug tracing is activated. The output is written to wherever `trace_output` is specified for that server.

In addition to the capability to set server stream output switches at startup, you can also dynamically alter the stream output switch settings for server processes that are already started. The ability to redirect output for a running server provides the capability to turn on a particular output stream (such as debug tracing) on a running server, capture the

output to a specified file, and then turn the stream output off after sufficient data has been captured. It also gives you the capability to capture stream output sessions to files using the `DMQ$MGR_UTILITY RO` (Redirect Output) option.

Tools for Troubleshooting BEA MessageQ Problems

BEA MessageQ is a complex, multiplatform, networked communication product. Its ability to function is affected by the computer network(s) over which it communicates, the hardware and software of each platform running BEA MessageQ, the configuration of each BEA MessageQ group, and the application environment for each application using BEA MessageQ.

Problems can occur in following areas:

- BEA MessageQ startup, including:
 - COM Server and link driver
 - MRS Server
 - SBS Server
 - Journal Server
 - Event Logger
 - CLS Server
 - NA Server
 - Qtransfer Server
- BEA MessageQ applications
- Message delivery
- Proper handling of undeliverable message actions (UMAs)
- BEA MessageQ resource depletion
- DECnet links

- BEA MessageQ process abortion
- BEA MessageQ performance
- BEA MessageQ troubleshooting tools fall into the following categories:
 - Server log messages
 - API return status values
 - Utility programs

Server Log Messages

Each BEA MessageQ Server logs informational messages to describe events or situations affecting BEA MessageQ. Each message provides a description of an event, reports on a condition of a BEA MessageQ component, or provides information regarding the operation of BEA MessageQ. Appendix E, “Error Log Messages,” contains a description of warning and error messages logged by the COM Server, SBS Server, MRS Server and link drivers.

The source of many BEA MessageQ processing problems can be found by viewing the contents of the server log files. For example, the COM Server log file can contain messages describing problems that may be encountered with group startup. Of particular importance is the group-wide event log file (DMQ\$EVL_bbbb_ggggg.LOG) which contains messages on problems that may affect the entire message queuing group.

For example, a failure to send a message to a remote node would appear in the event logger as:

Listing 12-2 Sample Log File Information for Message Failure

```
.  
. .  
EVENT_LOGGER      30-DEC-1997 14:00:19.12 I Message Undeliverable  
EVENT_LOGGER      30-DEC-1997 14:00:19.14 I + Discarded msg - Src=4.4  
Tgt=2.155  
                  Class=1 Type=-100  
EVENT_LOGGER      30-DEC-1997 14:00:19.16 I + Org-Src=2.2 Org-Tgt=4.4
```

12 Managing a BEA MessageQ Environment

```
Size=10
      Seq=00020002:00000C32
EVENT_LOGGER 30-DEC-1997 14:00:19.16 I %PAMS-E-NOTACTIVE, Target
process is not currently active - message not sent
```

API Return Status Values

BEA MessageQ applications will receive a return status value for each call made to BEA MessageQ API routines. Programmers should check the return status after calling API routines to perform the necessary error recovery steps. For a list of return status values for each BEA MessageQ API routine and a description of their meaning, refer to the *BEA MessageQ Programmer's Guide*.

Utility Programs

BEA MessageQ provides the following utilities to assist in troubleshooting system problems:

- DMQ\$TEST
- DMQ\$LOOP
- DMQ\$LLS_VERIFY
- DMQ\$MONITOR
- DMQ\$MGR_UTILITY
- DMQ\$SCAN_SYSTEM_FOR_DMQ.COM

The following describes how each utility can be used to troubleshoot problems. See Table 9-1 for information on running these utilities from the BEA MessageQ main menu.

DMQ\$TEST

The DMQ\$TEST utility allows you to attach to a queue, send and receive messages, and detach from the queue. This utility can be run directly or executed by choosing Option 6 from the BEA MessageQ main menu.

DMQ\$LOOP

The `DMQ$LOOP` utility allows you to test communication on the message queuing bus by performing a loopback test of messages. The utility can be run directly or by choosing Option 4 from the BEA MessageQ main menu.

DMQ\$LLS_VERIFY

`DMQ$LLS_VERIFY` allows you to check BEA MessageQ global sections. It is used to look for queues with large numbers of pending messages. This utility reports information about the buffer pools such as the number of pending messages and the parameter settings. The utility can be run directly or by choosing Option 3 from the BEA MessageQ main menu.

DMQ\$MONITOR

The `DMQ$MONITOR` utility allows you to view queues, queue counters, queue quotas, links to other groups and cross-group connections. Some monitoring can be performed dynamically in continuous mode. In addition, the utility provides the capability to reset queue counters, kill the COM Server, and set and alternate group. The utility can be run directly or by choosing Option 5 from the BEA MessageQ main menu.

DMQ\$MGR_UTILITY

The `DMQ$MGR_UTILITY` utility allows you to perform several BEA MessageQ management functions. For troubleshooting purposes, this utility offers the capability to set or redirect the stream output and trace switches for processes attached to the group including the server processes. This functionality is obtained by selecting the `RO` menu item from the `DMQ$MGR_UTILITY` utility. The utility can be run directly or by choosing Option 7 from the BEA MessageQ main menu.

DMQ\$SCAN_SYSTEM_FOR_DMQ.COM

This utility scans processes on the system and indicates which processes are MessageQ processes (processes that are using either `DMQ$ENTRYRTL` or `DMQ$EXECRTL`). This utility is run directly from the command line.

Troubleshooting Procedures

This section describes the basic tasks in troubleshooting BEA MessageQ and explains how to solve some of the most common problems.

Basic Troubleshooting Tasks

You can monitor BEA MessageQ output and isolate operational problems by directing and redirecting output streams, switching specific types of stream output on and off, and examining log files.

If you are having difficulty starting a BEA MessageQ group, look at the COM Server log file. Because the COM Server process is central to BEA MessageQ communications, it often contains informative messages that will help you to pinpoint reasons why the message queuing group will not start and why other communications failures occur.

Diagnosing Application Errors

To isolate a problem that occurs during application development, it is helpful to see the flow of BEA MessageQ routines. By defining the value of the `DMQ$DEBUG` logical name to `ALL`, you can view informational messages that document the execution of BEA MessageQ API routines. Setting the `DMQ$DEBUG` logical name to `ALL` will also cause debugging trace information to be written to wherever `DMQ$TRACE_OUTPUT` is directed.

These attributes can be set dynamically while the application is running using the `DMQ$MGR_UTILITY` program and specifying the `RO` (Redirect Output) option. Use this option to turn a switch on or off, and start and stop log file output as you need.

Enabling Server Tracing

Just as you can turn on tracing information for an application process, you can turn on tracing for a server process. In addition, servers allow a special form of tracing called server tracing. This causes the server to report server-specific processing during its operation. See “BEA MessageQ Server Logging/Debugging” for instructions on how to activate server tracing.

Verifying BEA MessageQ Group Startup

During the startup process for each message queuing group, BEA MessageQ:

- Creates the group logical name table
- Installs the necessary BEA MessageQ images
- Creates and loads the required global memory sections
- Starts up the COM Server process
- Creates and configures the queues, group tables, cross-group connections and other group configuration items and indicated in `DMQ$INIT.TXT`
- Creates the Event Logger and any other server process as indicated by entries in `DMQ$INIT.TXT`
- Ensures that all required servers are up and running

The startup procedure writes messages to `SYS$OUTPUT` to provide startup progress information. The last portion of a successful startup reports that all servers responded. BEA MessageQ Server processes are OpenVMS detached processes.

Note that these processes were created using the OpenVMS `SHOW PROCESS` command. The BEA MessageQ Server processes appears with a process name of `DMQ_c_bbbbgggg`, where *c* represents the server identifier, *bbb* represents the bus id, and *gggg* represents the group id. The server identifications are:

- C = Com Server
- L = Event Logger
- J = Journal Server

12 Managing a BEA MessageQ Environment

- M = MRS Server
- Q = Qtransfer Server
- S = SBS Server
- N = Naming Agent Server
- T = TCP/IP Link Driver
- D = DECnet Link Driver
- CLS_D = Client Lib Server (DECnet)
- CLS_T = Client Lib Server (TCP/IP)
- CLS_S = Client Lib Server (Single-Mode)

For example, the output of the `SHOW SYSTEM` command lists the COM Server, Event Logger, Journal Server, MRS Server, Qtransfer Server, and SBS Server processes:

Listing 12-3 Sample Output of SHOW SYSTEM Command

```
$
$ show system
OpenVMS V6.2 on node NODE1 9-JAN-1998 11:44:05.45 Uptime 108 03:00:49
  Pid Process Name State Pri I/O CPU Page flts Pages
20A00081 SWAPPER HIB 16 0 0 00:04:48.03 0 0
20A00086 CONFIGURE HIB 10 32 0 00:00:00.07 235 192
.
.
.
20A108A0 DMQ_C_009900040 LEF 8 16348 0 00:00:32.73 31491 6723
20A10BA5 DMQ_L_009900040 LEF 8 514 0 00:00:01.29 2336 1438
20A129B6 DMQ_T_009900040 LEF 8 144 0 00:00:00.88 2796 1787
20A141B7 DMQ_D_009900040 LEF 8 88 0 00:00:00.67 2391 1504
20A148B9 DMQ_J_009900040 LEF 7 180 0 00:00:01.43 5418 1827
20A12ABA DMQ_M_009900040 LEF 7 140 0 00:00:01.48 7225 2989
20A0E8BB DMQ_S_009900040 LEF 7 84 0 00:00:01.20 4136 1228
20A06F40 DMQ_N_009900040 LEF 7 1303 0 00:00:03.65 2689 1648
20A11B1F DMQ_Q_009900040 LEF 5 84 0 00:00:00.71 2454 1204
20A0F0C4 DMQ_CLS_D_6000 LEF 7 90 0 00:00:00.67 2248 1204
20A106CB DMQ_CLS_T_5001 LEF 7 112 0 00:00:00.68 2445 1355
.
.
.
```

To verify that the group has successfully started, you can run the `DMQ$TEST` utility. Attach to a temporary queue, send a message, read the message and then exit to verify that the group has started and is operational.

Troubleshooting BEA MessageQ Startup Problems

If the group failed to start, you need to isolate the problem and apply corrective action. Answer the following questions to help you pinpoint the source of the group startup failure.

Group Startup Failure

- Did the group successfully start? If not, did the startup procedure indicate there was a problem? If so, were there any accompanying instructions? For example, the following contains a portion of a failed attempt to start a BEA MessageQ group:

```
@$2$dka0:[dmq$v50.exe]dmq$startup 44 42
DMQ$STARTUP.COM - 14-DEC-1999 14:27:24.78

      Bus: 0044
      Group: 00042
      Disk: DISK$REDOAKDKA0:
      Timeout: 120
Start Servers: Yes
DMQ Version: DMQ$V50
  User Area: DMQ$DISK:[DMQ$V50.USER.0044_00042]
  Log Area:  DMQ$DISK:[DMQ$V50.LOG.0044_00042]

%DMQ-S-SETLNM, Set to MessageQ LNM table DMQ$LNM_0044_00042
%DMQ-W-NOGMTLNM, DMQ$GMT_OFFSET not set

Copyright ) BEA Systems, Inc. 2000. All rights reserved.
%DMQ-I-FORCEX, Forcing image termination on process "DMQ_C_004400042"
%DMQ-S-IMGTERM, Process "DMQ_C_004400042" image terminated
...Getting list of installed images
...Verifying DMQ$MSGSHR
...Verifying DMQ$EXECRTL
...Verifying DMQ$ENTRYRTL
```

12 Managing a BEA MessageQ Environment

```
...Verifying DMQ$VPS_EXEC
...Verifying DMQ$SET_LNM

%DMQ-S-SETLNM, Set to MessageQ LNM table DMQ$LNM_0044_00042
...Starting DMQ_C_004400042 (COM Server)
%RUN-S-PROC_ID, identification of created process is 21003190
***** DMQ_C_004400042 HAS NOT INDICATED IT'S UP AFTER 60 SECONDS *****
        - This maybe related to system load -
        Please examine the newly created version of DMQ$COM_SERVER_0044_00042.LOG
in DMQ$LOG. This file will most likely contain the reason for the
COM Server's failure to boot.
```

However, should this file not contain enough information there is another course of action. That is to run the COM Server interactively with trace turned on so that all errors are displayed on your terminal. This method requires that your process has sufficient quotas and privileges.

Run the COM Server interactively [Y/N] (Y)? Y

In the preceding example, the startup procedure indicated that there was a problem. It also instructs you to examine the COM Server log file. The COM Server may also be run interactively at this point to view the cause of the problem directly. If your process has sufficient quotas and privileges, enter Y at the prompt above.

```
.
.
.
$
$!***** define/user dmq$debug all
$ define/user dmq$script no
$ run dmq$exe:dmq$com_server
DmQ T 30:46.8 Time Stamp - 14-DEC-1999 14:30:46.86
DmQ T 30:46.8 ----- COM Server Starting -----
DmQ I 30:46.8 COM Server (E5.0-20(E2)) starting at 14-DEC-1999 14:30:46
DmQ I 30:46.9 MCS already exists - two DMQ COM servers running
%NONAME-F-NOMSG, Message number 00000004
$
$ v = f$verify(v)
%NONAME-F-NOMSG, Message number 00000004
%DMQ-F-STARTFAIL, Failed on startup of DMQ$COM_SERVER
%NONAME-F-NOMSG, Message number 00000004
$
```

The preceding listing indicates that two COM servers are running on the system. This means that the BEA MessageQ global memory section was not removed when the prior instance of the BEA MessageQ group was running. The cause of this problem is a process is still attached to the global memory section.

OpenVMS cannot remove the global section until all processes that reference that section have exited.

To locate the process that is still attached to the MessageQ global section use the provided command procedure: `DMQ$EXE:DMQ$SCAN_SYSTEM_FOR_DMQ` as follows:

```
$ @dmq$exe:DMQ$SCAN_SYSTEM_FOR_DMQ.COM
%DMQ-I-BLDLIST, Building scan list
%DMQ-I-SCAN, Scanning system for MessageQ processes
%DMQ-I-DMQPROC, Process="MQ_XYZ" PID=21003175 attached to MessageQ V5.0
%DMQ-I-SCANTOTAL, Found 1 processes using MessageQ
```

Once the process `MQ_XYZ` exits, the BEA MessageQ global section is removed from the system, and the BEA MessageQ group can be restarted.

Additional Group Startup Problems

The following questions can also help you identify the source of the group startup failure.

- Did the COM Server log file tell you why it couldn't start? The COM Server oversees the entire group. If a problem exists with the COM Server startup, the entire group will be affected.
- If the COM Server started, was there a problem with any of the other servers? Look in the event log file for event log file called `DMQ$EVL_bbbb_gggg.LOG` for information about which component may have had problems.
- If there is no event log file, is the disk full? Did the Event_Logger start? Execute the OpenVMS `SHOW SYSTEM` command and look for the `DMQ_L_bbbbgggg` process.
- If the event log file indicated a problem with another server, does the server log file for the failed server indicate the cause of the problem.
- If there is no server log file, does the server process exist? If not, is the server entry in the Profile section of the `DMQ$INIT.TXT` file set to `YES`?

Troubleshooting Problems with Running Groups and Queues

A BEA MessageQ message queuing group is a collection of processes and shared resources. Many factors can affect group operation. One of the most common problems is the depletion of one or more resources.

For example, message queues have resources limits restricting the number of messages and the total amount of message buffer pool they can use at any one time. If a running queue fails to send or receive messages, check the queue counters and quotas with the `DMQ$MONITOR` utility.

Groups also have a limited number of resources. For example, a group can only store a limited number messages as configured in the buffer pool configuration section of the `DMQ$INIT.TXT` file. If you experience group-wide failure or performance degradation, you should check the buffer pool memory.

The `DMQ$LLS_VERIFY` utility can assist in determining what amount of global memory sections are used and how much is available. If you are approaching the maximum for one or more BEA MessageQ resources, adjust the settings for the particular resource in the `DMQ$INIT.TXT` file and restart the group. Refer to Chapter 11, “Sizing and Tuning the BEA MessageQ Environment,” for information on setting OpenVMS and BEA MessageQ resource parameters.

BEA MessageQ Connectivity Troubleshooting

To deliver a message to its target queue, it must flow through the following components of the BEA MessageQ environment:

- Sender application
- Sender application's queue
- Local group's link driver input queue (for cross-group communication)
- Network
- Remote group's link driver input queue
- Receiver application's input queue

- Receiver application

Failure of any component along this chain will result in failed message delivery. Answer the following questions to isolate and correct a communications failure:

- Is the sender's application up and running properly? To determine this, Trace through the code (define `DMQ$DEBUG` to `ALL`) and/or use the OpenVMS debugger. Is the application sending the message? Is the function `pams_put_msg` reporting success? If not, what does the return status say? If not, is the application properly handling the failure?
- If the return status says that the message was sent, is BEA MessageQ actually sending it? Use the `DMQ$MONITOR` utility to determine if the message was put in target queue (if the target is local) or the link driver's queue (if the target is remote). To see the message as it is processed within the local group, use the `DMQ$MGR_UTILITY RO` (Redirect Output) function to turn on server tracing and header tracing for the COM Server and send it to the event logger. With this you can see if the message is processed by the COM Server by examining the event log file.
- If the target is a remote queue, has the message left the (local) link driver's queue?
- If not, are cross-group connections (`XGROUP_ENABLED`) enabled in the `DMQ$INIT.TXT` file? If not, enable this parameter. Also check the setting of `XGROUP_VERIFY`. If `XGROUP_VERIFY` is set, communication between groups will be prohibited unless the remote group is listed in the cross-group connections table.

If cross-group verification is enabled, check to see if the remote group has an entry in the cross-group connection table and, if so, that the group numbers, node names and group parameters for the remote node are correct.

If they are not, make the necessary corrections. Also verify that there is an entry in the cross-group table for the local group. After making the necessary changes to the `DMQ$INIT.TXT` file, you will need to restart the group.

- If cross-group connections are enabled but the message did not leave the (local) link driver, is the network up and running? Can you reach the remote node? Use NCP, Ethernim, OpenVMS `Set Host`, DEC TCP/IP or any other network tool to determine the operational status of the network.
- If you can reach the remote node, is BEA MessageQ up and running on the remote node?

- If it is, has the message reached the (remote) link driver's input queue? Use the `DMQ$MONITOR` utility, COM Server tracing and header tracing on the remote node to follow the message flow through the group.
- If the message passed through the link driver, has it reached the input queue for the receiving application. Is the application up and running? Is it attached to the group? Use the `DMQ$MONITOR` utility to determine if the message was input and is pending in the receiver's queue.
- If the application is attached, has it read the message?
- If it has read the message is it processing the message properly? Use the same tracing and debugging tools on the receiving application as before.

Due to production constraints, not every application can be tested using the actual application, queue or messages. For those situations, BEA MessageQ provides test utilities such as `DMQ$TEST`. This will allow you to attach to a test queue and send a test message to a target queue.

You can use the same tools (the `DMQ$MONITOR` utility, server tracing, logging, and network tools) with the `DMQ$TEST` utility to test end-to-end connectivity. We recommend that you have a test environment designed and ready where you can rapidly troubleshoot problems when they occur. Your test environment should include:

- Description of the test procedures and how to execute them
- Test applications for sending and receiving messages. This can be the `DMQ$TEST` utility, another test utility or a test version of your application.
- Message queues configured only for testing purposes
- Predefined test messages
- A networked environment

BEA MessageQ Recovery Troubleshooting

When an application utilizes recoverable messaging, a message must flow through the following BEA MessageQ components:

- Sender application
- Sender application's queue

- MRS input queue
- MRS SAF file
- Local group's link driver input queue (for cross-group communication)
- Network
- Remote group's link driver input queue
- Remote groups MRS input queue
- Remote group's MRS DQF file
- Receiver application input queue
- Receiver application

To isolate a problem with a recoverable message, trace error and event logging for each of these components as well as the following:

- Are there any error message written to the local MRS Server or `EVL` log files? If so, what do they tell you about the reason for the error?
- What was the PSB delivery status and UMA status that was returned to the sending applications from the `pams_put_msg` service?
- After the message left the sender's queue, did it reach the local MRS Server's input queue? Use the `DMQ$MONITOR` utility to view this.
- Has the message been written to the local MRS Server's SAF file? This can usually be detected by the PSB delivery and UMA states values returned from `pams_put_msg`. However, you can look into the SAF file by using the `DMQ$MGR_UTILITY JC` (Journal Control) function to see if the message is in the SAF file.

To do this, you will need to first run the `DMQ$MGR_UTILITY` program and specify the `JC` (function). Next, you will select the SAF menu option. When presented with the SAF controls menu you must first close the file (select the `CL` menu option, and specify the queue in response to the prompts). You can then dump the contents of the SAF file (select the `DU` menu option) and respond to the prompts asking where the dump file should be written and what to include in the dump file. Note that the last prompt in the `DU` function asks you whether to go ahead with the dump. Enter `Y` (the default). Reopen the SAF file (SAF menu option `OP`) and exit `DMQ$MGR_UTILITY`. You can then type the dump file. Note

that while the SAF file is closed, any attempts to write to the queue's SAF file will fail and the action indicated by the sender's UMA will be taken.

- Is there an accumulation of messages into the SAF file or an accumulation of SAF files? The `DMQ$MGR_UTILITY JC` command can tell you if there are messages in a SAF file. Use the OpenVMS directory command on the `DMQ$MRS :` directory to look for files that begin with `DMQ_bbbb_ggggg` and have a `.SAF` file extension. If there is an accumulation, is the network running? SAF files will accumulate if the recoverable messages cannot be received by the remote MRS Server.
- If the message has reached the remote MRS Server, has it been written to the DQF file? Use the `DMQ$MGR_UTILITY JC` function to look at the queue's DQF file. Processing for DQF files is the same as SAF files described above except specify DQF instead of SAF at the `JC` submenu. If the message has been written to the DQF file, then look from the DQF file to the remote application for the failure.

BEA MessageQ Application Troubleshooting

BEA MessageQ reports the success or failure of BEA MessageQ operations by returning status values from calls to BEA MessageQ API calls and by placing values in the delivery status field of PAMS Status Block (PSB). For detailed information on obtaining and using BEA MessageQ status values refer to the *BEA MessageQ Programmer's Guide*.

BEA MessageQ applications should test the return status and perform the proper handling based on these values. To assist in the debugging of BEA MessageQ application, BEA MessageQ allows tracing to be turned on during application execution. Tracing writes messages to the destinations specified in `DMQ$TRACE_OUTPUT` (`SYS$OUTPUT` by default) that describe the execution path through BEA MessageQ code.

Application developers can take advantage of this by defining the `DMQ$DEBUG` logical name to the value of `ERROR`, `TRACE`, or `ALL` before running their application. When set to `ERROR`, messages are written to the trace stream when BEA MessageQ routines encounter an error.

When set to `TRACE`, messages are written as BEA MessageQ executes its API functions. When set to `ALL`, BEA MessageQ reports both trace and error information to the trace stream. `ALL` is the most commonly used setting for `DMQ$TRACE_OUTPUT` during application debugging. An example of trace output is as follows:

Listing 12-4 Sample Trace Output

```
$ define dmq$debug all
$ run DMQ$EXE:DMQ$TEST

0.0                DMQ$TEST - Bus 0099                9-JAN-1998 12:21:28.08

-- Using PAMS_ATTACH_Q --
Attach mode (Name=1,Number=2,Temp=3) [3]:
Q type (PQ/SQ/MRQ) [PQ]:

DmQ T 23:40.0 Time Stamp - 9-JAN-1998 12:23:40.04
DmQ T 23:40.0 Tracing/logging event flag number = 61
DmQ T 23:40.0 Entering PAMS_ATTACH_Q
DmQ T 23:40.0 Entering PAMS_COMMON_ATTACH
DmQ T 23:40.0 Entering INIT_PARAMS
DmQ T 23:40.0 (req_q_num:0, &q_num:0X7FE0E8E8, q_type:-200, q_name:""
DmQ T 23:40.0 q_name_len:0, adver_scope:-202, mode:-212, tmo:600)
DmQ T 23:40.0 Entering MAP_AND_CHECK_QUEUE
DmQ T 23:40.0 Entering PAMS_ATTACH_TO_DMQ_ENV
DmQ T 23:40.0 (Is_CS:0 Process name: "+_RTA1:" PID: 20A10C07, LOCKTMO:600)
DmQ T 23:40.0 Using event flag 60
DmQ T 23:40.0 Using event flag 59
DmQ T 23:40.0 DMQ$BLOCKING_CONFIRM == NO
DmQ T 23:40.0 Wakeup Mode = EXEC_MODE_AST
DmQ T 23:40.0 Connecting to message bus #0099 group #00040
.
.
.
```

In addition to monitoring the execution of a BEA MessageQ application, you may wish to observe the behavior of a group while an application is running. The `DMQ$MONITOR` utility program allows you to observe several features of a running group. One often used feature is the continuous queue counter display. This allows you to monitor the sending, receiving, and pending queue counters for queues in a group.

12 *Managing a BEA MessageQ Environment*

You can use this display to determine if a queue is sending out the proper number of messages to the correct queue by looking at the send count for the application's queue and the receive count for the target queue. In addition to queue counters, the `DMQ$MONITOR` utility allows you to observe several other facets of a running BEA MessageQ group. See Chapter 10, "Using BEA MessageQ System Management Utilities," for more information on the `DMQ$MONITOR` utility.

13 BEA MessageQ Security

This chapter describes the BEA MessageQ for OpenVMS security features that allow system managers to:

- Restrict read access to message queues
- Protect information stored in global sections
- Control network access for cross-group communications
- Enable/disable cross-group control messages that can effect group communications

Rights Identifiers Used By BEA MessageQ

There are two rights identifiers used by BEA MessageQ. The identifiers are created during installation by the kit installation procedure if they do not already exist.

In order to run the loader (`DMQ$EXE:DMQ$LOADER`), the user must have either VMS `OPER` privileges or be granted the `DMQ$MANAGER` rights identifier.

The following actions require either `OPER` privileges or the `DMQ$MANAGER` rights identifier:

- Running `DMQ$LOADER`
- Starting and stopping queues from Manager Utility
- Starting and stopping Client Library services from Manager Utility

The other identifier is `DMQ$OPERATOR`. The following actions require either `OPER` privileges or the `DMQ$OPERATOR` rights identifier:

- Link management commands (except for the link management Inquiry command)
- Kill command from DMQ\$MONITOR
- Log file controls from Manager utility
- Redirect output/trace from Manager utility
- Journal Controls from the Manager utility
- Defining access control on queues

See “Defining Access Control on Queues” for more information on the DMQ\$OPERATOR rights identifier.

Defining Access Control on Queues

To control queue access, the system manager must have an account that holds a DMQ\$OPERATOR rights identifier or the VMS OPER privilege. The system manager can limit which processes are allowed to read from a queue by setting the flag in the Check ACL column of the Queue Configuration Table in the DMQ\$INIT file to Y. When the flag is set to Y, BEA MessageQ checks a zero-length access control file called DMQ\$ACCESS:DMQ\$*bbbbggggqqqq*.DAT, where:

<i>bbbb</i>	is the 4-digit bus ID
<i>ggggg</i>	is the 5-digit group ID
<i>qqqqq</i>	is the 5-digit queue number

To read from a queue, a process requires read access to the queue's access control file. BEA MessageQ checks access control by determining if the process can perform an OPEN/READ operation on the DMQ\$ACCESS:DMQ\$*bbbbggggqqqq*.DAT file.

To set up access control for a queue, you must create the access control file before processes can access its associated queue. Use the CREATE command at DCL level to create the file and use the SET ACL command to give the user account read access to

the file. The logical `DMQ$ACCESS` is specific to each group and is defined in `DMQ$USER:DMQ$BOOT.COM`. The default value of `DMQ$ACCESS` is the BEA MessageQ group-specific directory referred to by the logical `DMQ$USER`.

For example, use the following commands to give user identification code (UIC) [345,333] access to queue 4, on group 1, bus 15:

```
$ CREATE DMQ$ACCESS:DMQ$00150000100004.DAT
$ Ctrl/Z

$ SET ACL/ACL=( IDENTIFICATION=[ 345,333 ],ACCESS=READ) -
_$ DMQ$ACCESS:DMQ$00150000100004.DAT

$ SET PROT=(S,O,G,W) DMQ$ACCESS:DMQ$00150000100004.DAT
```

Securing Readout of Permanent Queues

If `Check ACL` is set to `Y` on a primary or secondary message queue, BEA MessageQ validates whether a process is allowed to read from a queue when the process first attaches to the queue. If the queue is a multireader queue (MRQ), BEA MessageQ validates access at the first attempt to read from the queue.

Securing the Creation of Temporary Queues

If `Check ACL` is set to `Y` on the `TEMPORARY_Q` queue and the queue number is 0, then only those processes that can read `DMQ$ACCESS:DMQ$bbbbggggg00000.DAT` are allowed to create temporary queues. This protection works for processes that call the `pams_attach_q` function. However, securing access to temporary queues results in the disabling of the message interface to the COM Server to create temporary secondary queues.

Setting the Global Section Protection Mask

You can configure a protection mask to restrict access to BEA MessageQ global sections. When the protection mask is configured, an access control list (ACL) can be applied to control individual user access. All BEA MessageQ global sections use one protection mask and, therefore, each requires an ACL to be set after the COM Server is booted.

BEA MessageQ uses system global sections, which are owned by UIC [1,4]. If the owner has not been granted access, the COM Server will be unable to initialize the global sections.

Defining Protection Mask Logical Name

The logical name controlling the global section protection mask definition is called `DMQ$SET_GBLSEC_PROT`. Define the logical name using the following command:

```
$ DEFINE/TABLE=DMQ$LNM_TBL/EXEC DMQ$SET_GBLSEC_PROT sogw
```

where:

- The option *sogw* corresponds to a YES/NO setting of access for system, owner, group, and world.
- All four settings are mandatory.

Table 13-1 shows typical combinations of protection settings.

Table 13-1 Sample Settings for `DMQ$SET_GBLSEC_PROT`

Setting	Description
YYYY	Default; unrestricted access to global sections
NYNN	Access only for BEA MessageQ Server processes and ACLs
NNNN	Access only through ACLs

Setting ACLs for Global Sections

When you set the ACL for the global sections, you must apply the ACL to all five global sections shown in Table 13-2. Set an ACL for a global section (called, in this case, `DMQ$MCS_C_bbbb_ggggg`) as follows:

```
$ SET ACL/OBJ=SYSTEM_GLOBAL_SECTION/ACL=( ID=___, AC=___ ) -
_ $ DMQ$MCS_C_bbbb_ggggg
```

Note: You must set the ACL after the COM Server is running. For example, you can add the ACL command to `DMQ$BOOT.COM` after the line that executes `DMQ$COM_START.COM`.

Table 13-2 shows the names for the five global sections where *bbbb* is the 4-digit bus ID and *ggggg* is the 5-digit group ID.

Table 13-2 BEA MessageQGlobal Section Names

Section Name	Description
<code>DMQ\$MCS_C_bbbb_ggggg</code>	Control section
<code>DMQ\$LLS_S_bbbb_ggggg</code>	Small buffer pool section
<code>DMQ\$LLS_M_bbbb_ggggg</code>	Medium buffer pool section
<code>DMQ\$LLS_L_bbbb_ggggg</code>	Large buffer pool section
<code>DMQ\$GNT_C_bbbb_ggggg</code>	Group Name Table (GNT) section
<code>DMQ\$GRP_C_bbbb_ggggg</code>	Cross-group control section
<code>DMQ\$MRQ_C_bbbb_ggggg</code>	Multireader queue (MRQ) control section

Controlling Network Access to Queuing Group

The Link Drivers can limit incoming cross-group connections to the nodes found in the cross-group connection table. The incoming cross-group connections are controlled by the `XGROUP_VERIFY` parameter in the Profile section of the `DMQ$INIT.TXT` file (see Chapter 3, “Configuring Cross-group Connections,” for more information).

When `XGROUP_VERIFY` is enabled, the cross-group connection table functions use an ACL to validate all connections against a known list of valid nodes and BEA MessageQ group IDs. If a connection fails to match exactly, that connection is dropped and a security match failure event is logged.

14 Managing Failover

Failover in a distributed system means the dynamic replacement of service after a system or component failure. This section presents an overview of failover and BEA MessageQ failover features:

- Failover Mechanisms
- Cold Failover
- Hot Failover to a Running Shadow Group
- Failover of Recoverable Messaging to and from a Single Target

Failover Mechanisms

BEA MessageQ for OpenVMS has two mechanisms for quickly switching service from a failed component to a backup service point: group failover and recoverable queue failover.

Group Failover

Group failover involves movement of the location of a group from one network node to another. This can be accomplished using the following two methods:

- Cold failover, involving rebooting a group on a different network node

- Hot failover, involving the dynamic switching of files and network links to a backup group that is already running but is in a unconnected state on another network node.

Immediately upon startup, a group establishes network links to all the groups listed in the cross-group connection table. If the first node for a group is not available, a connection is tried to the second node that is listed, and so on.

If no connection can be made to a group, it is marked as DOWN until a connection is made. After the group is started, connections to disconnected groups are retried periodically. This scheme allows clients to transparently establish a connection to a backup node when the primary node fails. (Messages sent to the group while it is unavailable will take the undeliverable message action (UMA).)

Recoverable Queue Failover

The recoverable queue failover involves movement of the recoverable journaling stream associated with the specific queue. This can be accomplished using the failover mechanisms shown in Table 14-1.

Table 14-1 Recoverable Queue Failover Mechanisms

Failover Mechanism	Description
DQF Redirection	The entire DQF stream for a queue is moved to another queue. This queue may be served in the same group or in a different group.
DQF Queue Merge	The messages in a DQF journal stream can be merged into another DQF journal stream.
SAF Redirection	Same as DQF redirection but involves store and forward files of the sending group.
SAF Queue Merge	The messages in a SAF journal stream can be merged into another SAF journal stream.

Managing and Planning for Failover

When a system running message queuing groups goes down, the system manager can bring the groups up on a backup node and then bring up the connected processes. When failover is needed, it is best if it is planned and designed early in the application design cycle. It can be difficult to back fit failover handling once an application is in production.

All the failover mechanisms require operational planning. In most cases, failover handling requires that an application code be written to support it.

The physical media (where recoverable journals reside) must be accessible to both the primary node where the application is originally running and the backup node where the application will failover to. In a VMS cluster, this can be rather a simple task because the access to disks can span network nodes. Outside of a VMS cluster, this is still possible, by physically moving or copying the media.

Cold Failover

The cold failover from one node to another requires no special application coding, but has the drawback of being the slowest of the failover schemes, since both BEA MessageQ and the application must be completely restarted.

Due to the fact that the `%XGROUP` section of the `DMQ$INIT.TXT` file allows several nodes to be specified as connection points for a group, when an attempt fails to connect a group to a node, BEA MessageQ attempts to connect to the next node in the list. Connections are retried on either a timed basis or when an explicit command is given to force connections.

Refer to Chapter 3, “Configuring Cross-group Connections,” for more information on cross-group configuration.

If one node is down, the group may be moved to the next node in the connection list and started as shown in Figure 14-1.

In the case where the two nodes do not share the same file system, the entire group environment needs to be physically moved before failover can begin. In a VMS cluster environment, all the nodes in the connection list can share files. In that case, failover from one node to another is accomplished by simply starting the group on the new node (as shown in Figure 14-2).

It is possible to automate this failover procedure with appropriate DCL command procedures.

Figure 14-1 Cold Failover For Non-clustered Environment

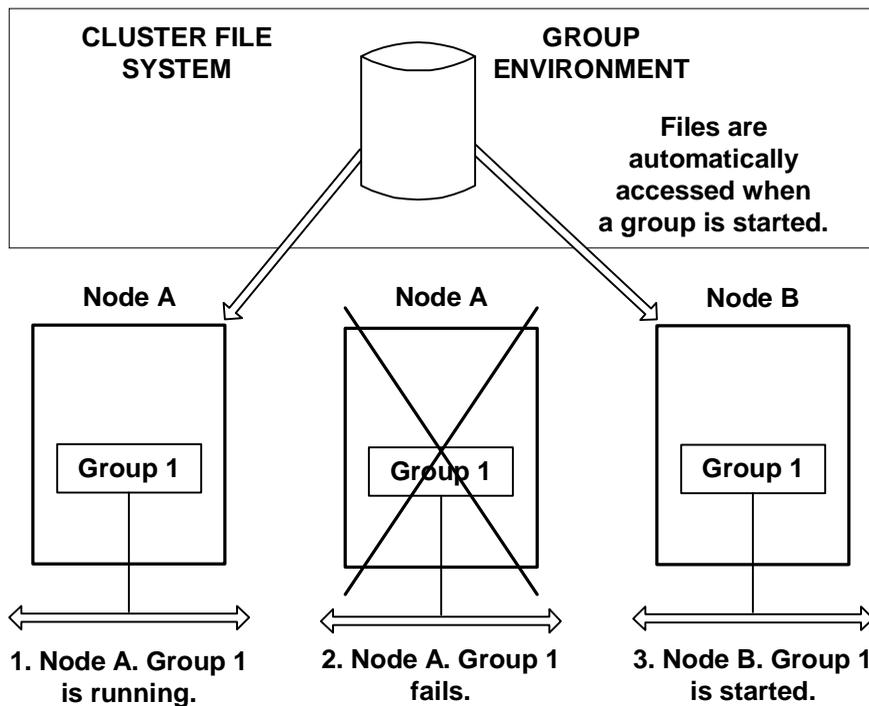
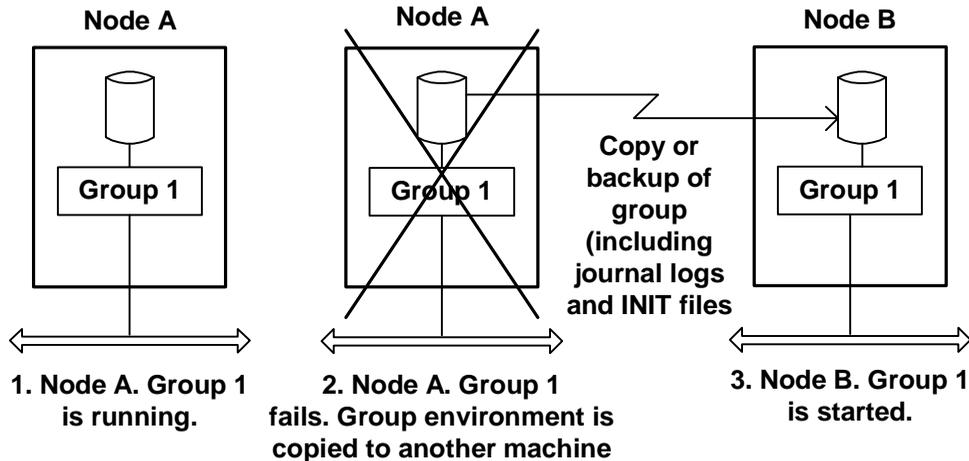


Figure 14-2 Cold Failover For Clustered Environment



When failing over to a standalone system (a system not part of a VMS cluster), submitter history checking on the receiving system should be disabled using the `DMQ$MRS_DISABLE_SH` logical name if the following conditions exist:

- system times are not synchronized; specifically, the system time on the standalone failover node is running behind the time on the primary node
- recoverable messaging to a Store and Forward (SAF) file is enabled

Submitter history checking does not need to be disabled if the failover system is part of a VMS cluster.

Hot Failover to a Running Shadow Group

Hot failover can be configured to be automatic, or can be managed by a watchdog process.

Automatic Synchronized Cluster Failover

In automatic synchronized cluster failover, a customer (for example, a system manager) maintains a running copy of the group on each of the backup nodes. Only one of the groups is the primary group. The primary group holds a cluster lock which prevents the other secondary (backup) groups from starting any BEA MessageQ servers other than the COM Server and Event Logger. When the primary group goes down for any reason it will release the cluster lock and one of the secondary groups will obtain the lock and become the primary group. After a secondary group becomes the primary it will startup all of its servers. A secondary group can perform local messaging while waiting to become the primary group, but it cannot perform any cross-group messaging.

Implementing Automatic Synchronized Failover

Automatic synchronized cluster failover is enabled by default via the logical `DMQ$GROUP_SYNCHRONIZE` which is predefined in the `COM` section of the `DMQ$SET_SERVER_LOGICALS.COM` file. When this logical is set to `YES`, or not defined at all, a cluster lock is used by BEA MessageQ to determine which group is the primary group. Therefore, a group may be started on multiple nodes within a cluster. Only one of the groups (first one up) may hold the cluster lock. A group cannot become the primary group until it gains the cluster lock. The cluster lock name is `DMQ$PRIMARY_GRP_bbbb_ggggg`, where `bbbb` is the bus number and `ggggg` is the group number.

If a group cannot obtain the cluster lock it will log the following message in the `EVL` log.

```
COM_SERVER          9-JAN-2000 12:03:10.77 I This group is waiting to
become the primary
```

When a group obtains the cluster lock it will log the following message in the `EVL` log.

```
COM_SERVER          13-JAN-2000 23:48:50.22 I This group is the primary
```

If a group is not the primary group, it will only start up the `COM` server and Event Logger. This is important because the primary and backup groups:

- cannot share the network simultaneously if defined with the same group address.

- cannot access the journal files at the same time.
- cannot access the checkpoint file at the same time.

Non-automatic Hot Failover

In non-automatic hot failover, a customer (for example, a system manager) maintains a running copy of the group on another node, but it is set up in such a way that it has no cross group links and has no journal files opened. When a failover is needed, the cross group links from the failed group are terminated, and files are closed. (This may occur naturally in the event of a hardware failure on the machine). Then BEA MessageQ allows links from the shadow group to be set and files opened.

For both primary and backup nodes, the %XGROUP table must have the Gen Cnt field set to D (disabled). Listing 14-1 shows a sample of the cross-group connection table.

Listing 14-1 Sample Cross-Group Connection Table

```

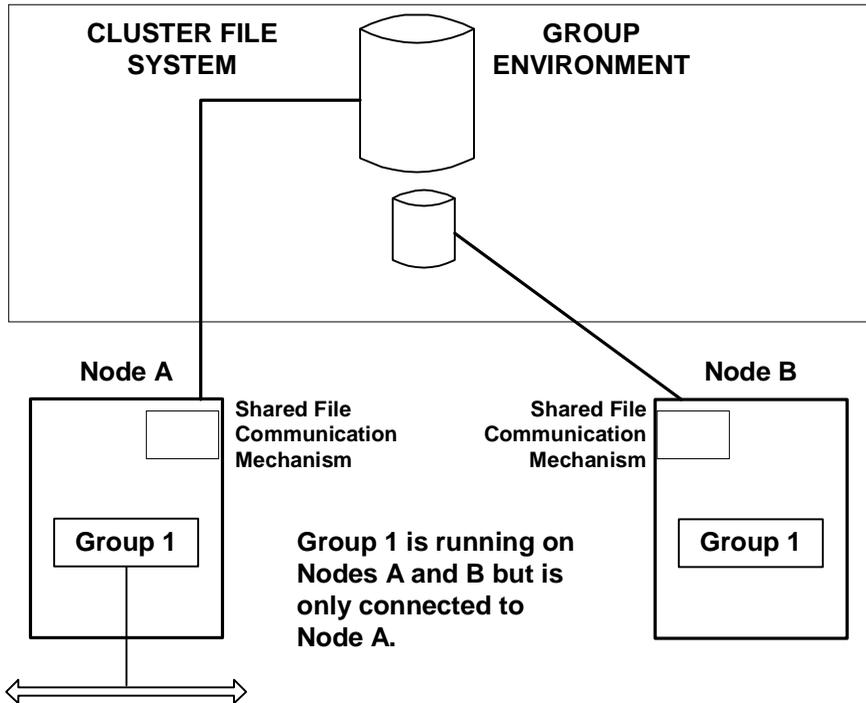
-----
%XGROUP      ***** Cross-Group Connection Table *****
*
*
*gname       gnumber   node          gen buf   buf  recon  dly win  transport
*            cnt warn  max         kb   kb    sec   sec msg  type  endpt
*-----
ONE          1     BIGVAX       Y  -1    75    0    -1  -1  DECNET
TWO          2     GVAX01       D  -1    -1    -1    -1  -1  DECNET
TWO          2     GVAX02       D  -1    -1    -1    -1  -1  DECNET
TWO          2     GVAX03       D  -1    -1    -1    -1  -1  DECNET
*
%EOS
-----

```

Figure 14-3 shows a normal condition when nodes share files on the cluster before a failover occurs.

Figure 14-3 Normal File Sharing for a Running Group

Implementing Non-Automatic Hot Failover

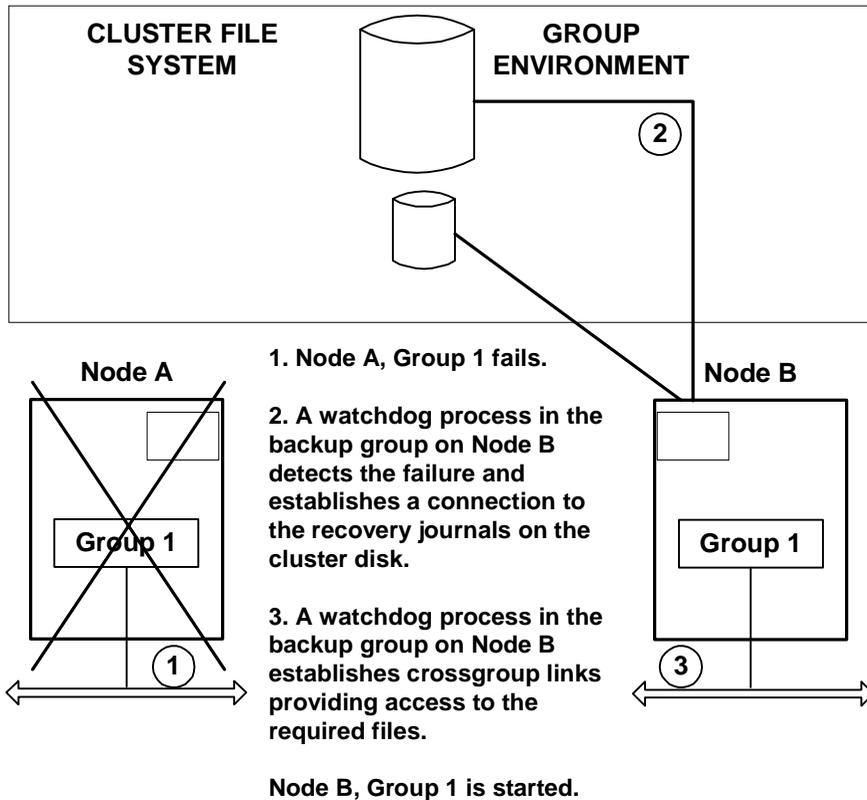


Any backup copy of a group must be started without any cross-group connections. This is important since two groups with the same group address cannot share the network simultaneously. The group links can be disabled using the `DMQ$INIT.TXT` file (XGROUP Table).

Failover is then accomplished by explicitly tearing down the links to one node (if they have not dropped as a result of the failure), and then forcing links to the next node in the failover list (as shown in Figure 14-4).

This procedure works for nonrecoverable messaging, but runs into difficulty if the backup group has message recovery services (MRS) enabled. The problem is that only one recovery service can own recovery journals (DQF, SAF, PCJ or DLJ file) at one time. The journals can not be shared between the original and backup group.

Figure 14-4 Hot Failover for a Running Group



The solution to this problem is to tell the recovery system of the backup group to start up but remain idle. This is done by setting the logical name `DMQ$MRS_DISABLE_JOURNALING` to `YES` in the `MRS` section of the `DCL` command procedure, `DMQ$USER:DMQ$SET_SERVER_LOGICALS.COM`.

The journal files may then be opened only when a `MSG_TYPE_MRS_JRN_ENABLE` message is sent to the `MRS` Server indicating that it should open the journals. Failover of recoverable messaging from the primary to the backup group is completed by sending the `MSG_TYPE_MRS_JRN_ENABLE` message to the `MRS` Server in the backup group.

Detecting a Failure While the Failing Group Is Still Running

Some applications may want to initiate failover from the primary group to the backup group when the primary group is still running. The heuristic for handling this type of failover is rather complex and is often application dependent.

To close all journal files, the `MSG_TYPE_MRS_JRN_DISABLE` message must be sent to the MRS Server. Therefore, the way to initiate the failover is to send a `MSG_TYPE_MRS_JRN_DISABLE` control message to the MRS Server of the group that is failing. This tells MRS to close all journal files. `LINK_MANAGEMENT` control messages are then used to disconnect the primary group from the network.

A `MSG_TYPE_MRS_JRN_ENABLE` message is sent to the MRS Server of the backup group causing it to open the journals. Finally, the links to the backup group are brought up and messaging can continue.

Programming Considerations

Unless automatic synchronized cluster failover is used, significant programming effort is required for hot failover to be successful in a production environment. Usually, a watchdog program is coded to run on the primary and backup nodes. This program maintains information on the state of both primary and backup node and periodically decides whether the primary or backup node owns the processing.

A communication mechanism is needed between the watchdog programs. It may be as simple as a shared file that is polled periodically.

The heuristic for determining that a failover operation is required will vary according to the application. One scheme that works in a VMS cluster environment is to use the OpenVMS lock manager so that a lock is granted when some critical process dies.

Failover of Recoverable Messaging to and from a Single Target

Two methods for handling failover of a single Destination Queue File (DQF) are the DQF REDIRECT (Failover) option and the MERGE (QTRANSFER) option.

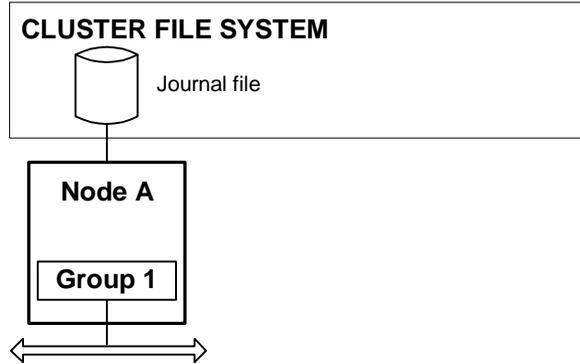
Redirecting Recoverable Data Streams

The DQF Redirect option works for redirecting an entire recoverable data stream from one queue to another queue (which may or may not be in the same group). The DQF journal must be closed before the failover can be attempted. If the link to the group is down, but the group is still running, you will need to explicitly force the recovery system to close the files. You can perform this failover dynamically using the RE option of `DMQ$MGR_UTILITY` journal control menu. Refer to Figure 14-5.

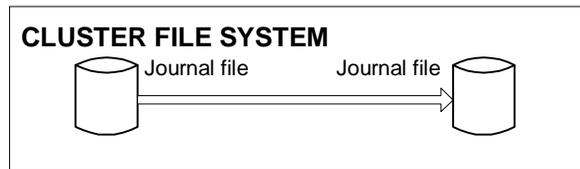
The entire DQF data stream must be on the same disk drive to be redirected (renamed) to a new DQF data stream. OpenVMS does not allow file renaming across devices.

Figure 14-5 DQF or SAF Redirecting for Failover

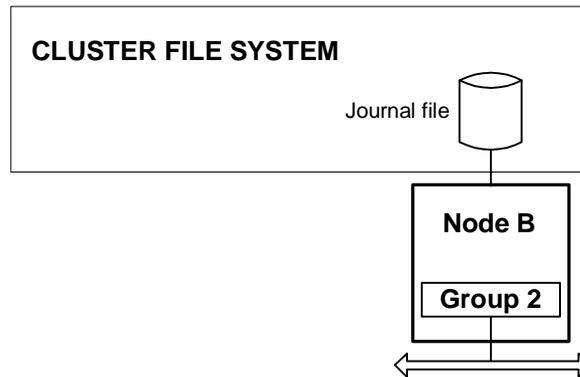
1. Data stream is opened on Node A, Group 1.



2. Data stream is closed, then renamed.



3. Data stream is opened on Node B, Group 2.



Merging Recoverable Data Streams

The MERGE option on the DQF screen allows you to merge a recoverable data stream with another data stream. The merging is controlled by the QTRANSFER Server that handles the moving of the contents of a DQF journal. This is accomplished by enveloping each message and sending the messages one by one to the MRS Server of the target group. The MRS Server then unpacks the envelope, writes (merges) the message to the DQF journal, and forwards it to the target queue.

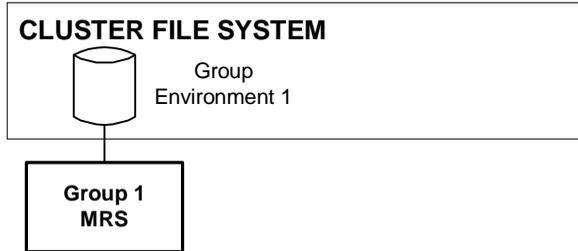
You can manage the merging option using the ME option on the DQF screen.

For the QTRANSFER process to work, the first DQF must be closed, either because the group serving the DQF is down, or because the DQF was explicitly closed using a control message. The QTRANSFER Server is then told to initiate the transfer by another control message. This is managed by a dialog between the QTRANSFER Server and the MRS Server.

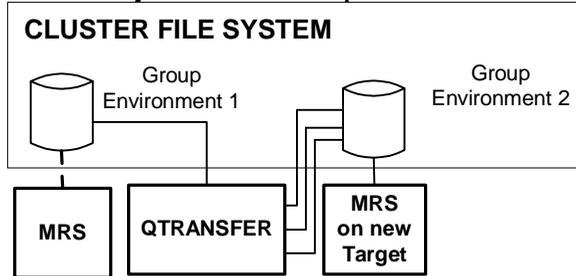
Figure 14-6 shows major steps in the MERGE process.

Figure 14-6 Data Stream Merging for Failover

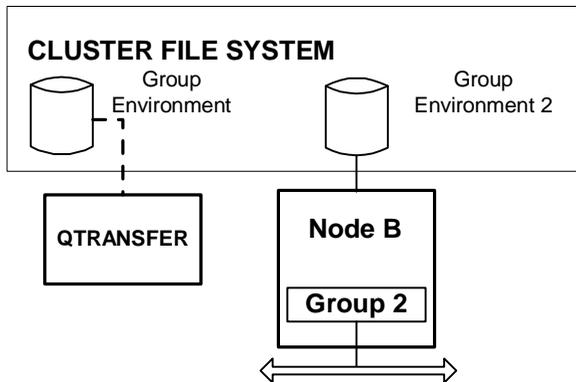
1. Data stream is opened on Node A, Group 1.



2. MRS is closed. QTRANSFER sends messages one by one to a new group disk. New messages are merged with other recoverable journals. MRS is opened on the new target.



3. Data stream is opened on Node B, Group 2.



Supporting Failover of SAF Files

The SAF failover works identically to the DQF failover. However, it may be a much more complex operation because the senders may be distributed across many systems.

In general, redirecting a SAF data stream involves:

- telling the sender applications to stop sending to a target
- closing the SAF file
- directing the MRS Server to pick up the SAF file and redirect it to another target
- telling the sender applications to start sending to the target

You can start the SAF failover using the Redirect (RE) option on the SAF screen.

15 Configuring the BEA MessageQ Bridge

BEA MessageQ V5.0 includes a messaging bridge that allows the exchange of messages between BEA MessageQ V5.0 and BEA TUXEDO V6.5. BEA MessageQ applications can send a message using `pams_put_msg` that a TUXEDO application can retrieve through a call to `tpdequeue`. TUXEDO applications can send a message using `tpenqueue` that a BEA MessageQ application can retrieve through a call to `pams_get_msg(w)`. In addition, a BEA MessageQ application can invoke a TUXEDO service using `pams_put_msg`. It is also possible for a TUXEDO application to use `tpenqueue` to put a message on a queue and to use `tpdequeue` to retrieve a message from a queue.

This exchange of messages is made possible by two TUXEDO servers that are included in the BEA MessageQ kit and that run on the same machine as BEA MessageQ: `TMQUEUE_BMQ` and `TMQFORWARD_BMQ`.

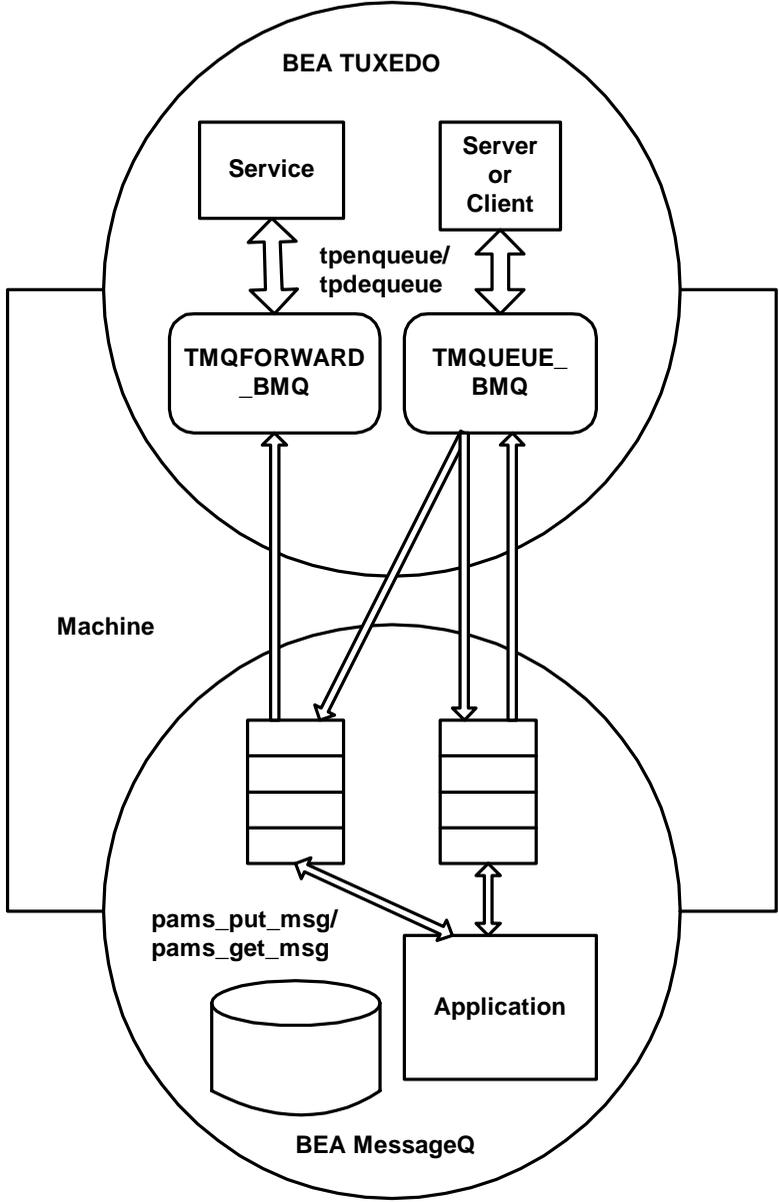
`TMQUEUE_BMQ` redirects TUXEDO `tpenqueue` requests to a BEA MessageQ queue where they can be retrieved with `pams_get_msg(w)`. `TMQUEUE_BMQ` also redirects `pams_put_msg` or `tepenqueue` requests to TUXEDO where they can be retrieved with `tpdequeue`.

`TMQFORWARD_BMQ` listens on specified BEA MessageQ queues and forwards `pams_put_msg` requests to a TUXEDO service. It also puts a reply or failure message on the sender's response queue.

The target queue and service are defined when `TMQUEUE_BMQ` and `TMQFORWARD_BMQ` are configured. This ensures that message exchange between BEA MessageQ and TUXEDO is transparent to the application.

Figure 15-1 illustrates message exchange between MessageQ and TUXEDO.

Figure 15-1 Message Exchange Between MessageQ and TUXEDO



Enabling the Messaging Bridge

The TMQUEUE_BMQ and TMQFORWARD_BMQ servers are part of the BEA MessageQ kit and are installed when BEA MessageQ is installed. During the installation procedure, you are prompted to choose one of the following installation options for BEA MessageQ and TUXEDO integration:

- install on top of BEA TUXEDO V6.5
- install without BEA TUXEDO

BEA TUXEDO must be installed and configured on the system where BEA MessageQ is being installed. The OpenVMS installation procedure requires that the TUXDIR logical is defined before the installation proceeds. If the TUXDIR logical is not defined, the following messages are displayed:

```
%VMSINSTAL-E-NOTUXDIR Cannot locate logical TUXDIR
%VMSINSTAL-E-NOTUXDIR MessageQ/TUXEDO bridge cannot be installed
```

If you choose to install on top of BEA TUXEDO V6.5, the applicable files for the TMQUEUE_BMQ and TMQFORWARD_BMQ servers are installed on your system. If you install without BEA TUXEDO, the TMQUEUE_BMQ and TMQFORWARD_BMQ servers are not installed on your system. See the installation and configuration documentation specific to your platform for detailed installation and configuration instructions.

Once the TMQUEUE_BMQ and TMQFORWARD_BMQ servers are installed, you enable message enqueueing and dequeuing for the application by specifying the servers as application servers in the *SERVERS section of the TUXEDO ubbconfig file. See the TMQUEUE_BMQ and TMQFORWARD_BMQ reference pages in the *BEA MessageQ Reference Manual* for detailed information on the server configuration syntax. Listing 15-1 shows a sample BEA TUXEDO ubbconfig file with the TMQUEUE_BMQ and TMQFORWARD_BMQ servers specified as application servers.

Listing 15-1 Sample BEA TUXEDO UBBCONFIG File

```
#Copyright (c) 1992 Unix System Laboratories, Inc.
#All rights reserved
#ident "@(#)apps:qsample/ubb.sample 60.5"
#
*RESOURCES
IPCKEY 52617
```

15 Configuring the BEA MessageQ Bridge

```
DOMAINID    qsample
MASTER     SITE1
MODEL      SHM
#
*MACHINES
#
"birch.beasys.com"
    LMID = SITE1
    TUXDIR = "$3$DKB400:[TUXEDO]"
    TUXCONFIG = "CYPRES$DKA200:[POPP.TUXTEST.QSAMPLE]TUXCONFIG"
    TLOGDEVICE = "CYPRES$DKA200:[POPP.TUXTEST.QSAMPLE]TLOG"
    TLOGSIZE=10
    APPDIR = "CYPRES$DKA200:[POPP.TUXTEST.QSAMPLE]"
    ULOGPFX = "CYPRES$DKA200:[POPP.TUXTEST.QSAMPLE]ULOG"
#
*GROUPS
#
GROUP1
    LMID = SITE1  GRPNO = 1
    TMSNAME=TMS TMSCOUNT=2
QUE1
    LMID = SITE1  GRPNO = 2
    TMSNAME = TMS_QM TMSCOUNT = 2
    OPENINFO = "TUXEDO/QM,CYPRES$DKA200:[POPP.TUXTEST.QSAMPLE]QUE,QSPACE"
#
*SERVERS
#
DEFAULT:    CLOPT="-A"
server     SRVGRP=GROUP1 SRVID=2
#
TMQUEUE_BMQ
    SRVGRP = QUE1  SRVID = 1
    GRACE = 0  RESTART = Y CONV = N MAXGEN=10
    CLOPT = "-s QSPACE:TMQUEUE -- -b 99 -g 51 "
#
TMQFORWARD_BMQ
    SRVGRP=QUE1    SRVID= 5 GRACE=0 RESTART=Y CONV=N MAXGEN=10
    CLOPT="-- -t 30 -q STRING -b 99 -g 52 " REPLYQ=N
*SERVICES
```

The `ubbconfig` file shown in Listing 15-1 can be used with the `qsample` sample application included in your BEA TUXEDO kit. No changes are required to the application. The client portion of the application (`client.c`) performs `tqueue`

TUXEDO Queue Space to BEA MessageQ Group Name

BEA MessageQ uses the `target` argument of the `pams_put_msg` function to specify the target queue address for a message. TUXEDO uses the `qspace` and `qname` arguments of the `tpenqueue` and `tpdequeue` functions to specify the target queue for a message

The TUXEDO queue space name must be the name of a service advertised by `TMQUEUE_BMQ` or `TMQFORWARD_BMQ`. The service name maps directly to a BEA MessageQ group. By default, `TMQUEUE_BMQ` and `TMQFORWARD_BMQ` automatically offer services named “`TMQUEUE_BMQ`” and “`TMQFORWARD_BMQ`” unless the `-s` command line option is specified. These default services map to the BEA MessageQ group to which they are attached, as specified by the `-g` command line option.

The function name to which services should be mapped is `TMQUEUE`. Each entry in the TUXEDO `ubbconfig` file for a `TMQUEUE_BMQ` or `TMQFORWARD_BMQ` server should be configured with a different alias for the default function name using the TUXEDO `-s` command line option. For example, one configuration of `TMQUEUE` may be named `Payroll`, while another is named `Sales`. This provides a way to precisely specify a BEA MessageQ entry point for a particular `tpenqueue` or `tpdequeue` call. If multiple instances of the same advertised service are running, TUXEDO performs load balancing and data dependent routing to determine which server handles the request.

Listing 15-3 shows a portion of a `ubbconfig` file containing multiple `TMQUEUE` configurations:

Listing 15-3 UBBCONFIG File with Multiple TMQUEUE Configurations

```
*GROUPS
TMQUEUE_BMQGRPHQMGR GRPNO=1
TMQUEUE_BMQGRPHQPLEBE GRPNO=2
TMQUEUE_BMQGRPREMOTENA GRPNO=3
TMQUEUE_BMQGRPREMOTEEUROPE GRPNO=4

*SERVERS
TMQUEUE_BMQ SRVGRP="TMQUEUE_BMQGRPHQMGR" SRVID=1000 RESTART=Y
  GRACE=0 CLOPT="-s Payroll:TMQUEUE -s
  Promote:TMQUEUE -- -b 5 -g 7"
TMQUEUE_BMQ SRVGRP="TMQUEUE_BMQGRPHQPLEBE" SRVID=1000 RESTART=Y
  GRACE=0 CLOPT="-s Payroll:TMQUEUE -s
```

```

Promote:TMQUEUE -- -b 5 -g 10"
TMQUEUE_BMQ SRVGRP="TMQUEUE_BMQGRPTEMOTENA" SRVID=2002 RESTART=Y
GRACE=0 CLOPT="-s Sales:TMQUEUE -- -b 5 -g 42"
TMQUEUE_BMQ SRVGRP="TMQUEUE_BMQGRPTEMOTEUROPE" SRVID=2002
RESTART=Y GRACE=0 CLOPT="-s Sales:TMQUEUE -- -b 12 -g 53"

*SERVICES
Payroll ROUTING="SALARYROUTE"
Payroll ROUTING="HAIRCOLORROUTE"

*ROUTING
SALARYROUTE FIELD=Salary BUFTYPE="FML32"
RANGES="MIN - 50000:TMQUEUE_BMQGRPPEBE,50001
-MAX:TMQUEUE_BMQGRPHQMR"
HAIRCOLORROUTE FIELD=Hair BUFTYPE="FML32"
RANGES=" 'Gray' :TMQUEUE_BMQGRPHQMR, *:TMQUEUE_BMQGRPPEBE"

```

In this example, three queue space names (Payroll, Promote, and Sales) are defined for two busses to four different BEA MessageQ groups (7, 10, 42, and 53). Two servers offer the same aliases (Payroll and Promote) with data dependent routing performed using the Sales and Hair fields respectively. The two other servers offer the same alias (Sales) with routing determined by load balancing and availability.

TUXEDO Queue to BEA MessageQ Queue

Any BEA MessageQ queue can be accessed by TUXEDO through the TMQUEUE_BMQ and TMQFORWARD_BMQ servers. However, BEA MessageQ queues are accessed in different ways depending on whether they are named or unnamed queues.

BEA MessageQ named queues can be local (group-wide) or global (bus-wide). To address a locally named queue from TUXEDO:

1. Configure the TMQUEUE_BMQ or TMQFORWARD_BMQ server to attach to the local group in which the named queue is defined.
2. Configure routing information to handle multiple instances of the TMQUEUE_BMQ or TMQFORWARD_BMQ server with the same alias as shown in “TUXEDO Queue Space to BEA MessageQ Group Name.”.

15 *Configuring the BEA MessageQ Bridge*

3. Use the queue name as defined by BEA MessageQ as the second parameter for `topenqueue` or `tpdequeue`.

To access an unnamed BEA MessageQ queue from TUXEDO, use an absolute queue identifier as the second parameter for `topenqueue` or `tpdequeue`. The absolute queue identifier is a combination of the BEA MessageQ group identifier and queue identifier formatted as *group_id.queue_id*. For example, queue 1005 in group 3 is specified as "3.1005". When accessing a queue in the local group, either specify the group as 0 or drop the group identifier and delimiter. For example, queue 1005 in the local group is specified either as "0.1005" or "1005". Queue identifiers that do not use this syntax, or are outside the valid range of group or queue numbers are assumed to be queue names.

A Sample DMQ\$INIT.TXT File

Listing A-1 is a sample of the MessageQ `DMQ$INIT.TXT` file that is used to customize the BEA MessageQ software.

Listing A-1 Sample BEA MessageQ DMQ\$INIT.TXT File

```

*****
*
*                               MessageQ Initialization File
*
*****
*
* VERSION SECTION
*
* The Version Section begins with %VERSION, and consists of a single line
* identifying the initialization file version. MQ V5.0 accepts init files
* with a version of 1.0, 2.0, or 3.0, 3.1, 3.2, 4.0
*
%VERSION 4.0
*
* PROFILE SECTION
*
* The Profile Section begins with %PROFILE, and contains lines defining the
* group-wide configurable limits. The values given below are examples, and
* do not represent "real" values.
*
%PROFILE      ***** Profile Parameters *****
*
ACCEPT_KILL_CMD      NO      ! Control MONITOR terminate requests
ENABLE_XGROUP        NO      ! Enable MessageQ cross-group access

```

A Sample DMQ\$INIT.TXT File

```
XGROUP_VERIFY          NO          ! Limit incoming cross-group connections
ENABLE_MRS             YES         ! Enable MessageQ Message Recovery Services
ENABLE_JRN            NO          ! Enable MessageQ Message Journaling Services
ENABLE_SBS            YES         ! Enable MessageQ Selective Broadcast Services
ENABLE_QXFER          NO          ! Enable MessageQ MRS Queue Transfer Services
FIRST_TEMP_QUEUE      200         ! Select start of temp queue pool (200-950)
XGROUP_TABLE_SIZE     25          ! Select max number of group entries (25-32000)
RCV_MSG_QUOTA_METHOD  MAX         ! Select rcv msg quota deduction (MIN | MAX)
ATTACH_TMO            600         ! Select PAMS_ATTACH_Q timeout (100-36000)
GROUP_MAX_USER_QUEUE  999         ! Maximum temp queue number (999-3999)
GROUP_MAX_MESSAGE_SIZE 128000     ! Maximum number of bytes per message
*DEFAULT_NAMESPACE_PATH /DMQNS/ ! Default namespace path for the Naming Agent
*
%EOS
*
*
* XGROUP SECTION
*
* The Xgroup Section begins with %XGROUP, and contains lines defining the
* cross group connections that will be established by this group. Each line
* defines a single cross-group connection. All parameters for a given
* connection must appear on the same line. A general explanation of each
* parameter is given below, followed by a an example cross group entry.
*
* GNAME - Name by which the remote MQ group is known to the local group.
*
* GNUMBER - MQ group number of the remote group.
*
* NODE - Name by which the remote node is known to the local node.
*
* INITIATE - "Y", "N", or "D" indicating whether connections to this node
* should be initiated (Y), not initiated (N), or disabled (D).
* A "Disabled" connection requires a link management command to
* enable connections to occur.
*
* THRESHOLD - Value for dynamic memory consumption at which congestion
* control will begin for this cross-group connection.
* (not used in VMS MQ V5.0)
*
* BUFFER - Maximum number of Kbytes that can be written to this link.
* POOL - This should exceed GROUP_MAX_MESSAGE_SIZE.
*
* RECONNECT - Interval, in seconds, between reconnect attempts when this
* cross group link is not connected.
*
* DELAY - Delay, in seconds, that a sender must wait before using a new
* window when the receiver is congested.
*
* WINDOW - Maximum transmission window size for this cross-group link.
```

```

*
* TRANSPORT - Network protocol stack used, either TCPIP or DECNET.
*             The DECNET transport is only available MQ for VMS.
*
* ENDPOINT   - For TCPIP transport, the internet port number of the remote
*             link listener process. For DECNET, this field should be blank.
*
* A sample xgroup table can be seen below:
*
%XGROUP      ***** Cross-Group Connection Table *****
*
*           gen buf   buf   recon  dly win  transport
*           cnt warn pool  sec    sec msg  type  endpt
*-----
*GR1_TCP      1 ipaddr1.company.com  Y  -1   75   -1  -1 -1 TCPIP  2001
*GR1_TCP      1 ipaddr2.company.com  Y  -1   75   -1  -1 -1 TCPIP  2001
*GR1_DECNET   1 daddr1                Y  -1   -1   -1  -1 -1 DECNET
*GR1_DECNET   1 daddr4                Y  -1   -1   -1  -1 -1 DECNET
*
*GR2_TCP      2 ipaddr3.company.com  Y  -1   75   -1  -1 -1 TCPIP  2002
*GR1_TCP      1 ipaddr3.company.com  Y  -1   75   -1  -1 -1 TCPIP  2002
*GR2_DECNET   2 daddr4                Y  -1   -1   -1  -1 -1 DECNET
*GR2_FAILOVER 2 daddr5                Y  -1   -1   -1  -1 -1 DECNET
*
*
%EOS
*
* ROUTING SECTION
*
* The Routing Section begins with %ROUTE, and contains lines defining the
* MessageQ static routing data base. All parameters for a given route must
* appear on the same line. The general format of a routing definition is
* given below, followed by an explanation of each parameter.
*
* TARGET_GROUP      - Group for which traffic must be routed.
*
* ROUTE_THROUGH_GROUP - Group to which traffic for the target_group must be
*                       sent.
*
%ROUTE
*
* Target      Route-Thru
* Group       Group
*-----
*      6             1
*     71            4
*
%EOS

```

A Sample DMQ\$INIT.TXT File

```
*
*
* CLS SECTION
*
* you can start any number of client library server processes (one per line)
* by enumerating each process and its endpoint below.
*
* ENDPOINT          defines the TCP/IP port number or DECnet object name
*                   that the server uses to accept incoming connections.
*                   TCP/IP - the value is limited to 1024-65535 inclusive.
*                   DECnet - the value is limited to 1-99999 inclusive.
*
* TRANSPORT         the name of the network protocol stack to be used for
*                   the server.  valid values are DECnet and TCP/IP.
*
* MAX_CLIENTS       the maximum number clients the server will allow at any
*                   given time.  Valid values are 1-512 inclusive.  VMS only.
*
* SECURITY_FILE      the full path name of the security file.  if no file
*                   is specified then a default file name is used.  see
*                   documentation for details.
*
%CLS      **** Client Lib Server Configuration Table ****
*
*
*   Endpoint          Transport          Maximum #      Security
*   -----          -
*   5000              TCPIP              16             dmq$user:cl_5000.sf
*   5001              TCPIP              16
*   6000              DECNET             32
*
*
%EOS
*
* GLOBAL MEMORY BUFFER POOL CONFIGURATION SECTION
*
* The buffer pool contains space for ALL messages on a VMS MQ V5.0 system.
* It must be sized to contain all messages active in memory at any given
* time.  When this pool is fully allocated, attempts to send a message cause
* PAMS_REMQUEFAIL to be returned to the sending application.  If the
* GROUP_MAX_MESSAGE_SIZE is larger than a LARGE buffer, then multiple
* buffers are chained together to contain such a message.  The maximum
* LARGE buffer size is 65K.
*
%BUFFER      ***** Buffer Pool Configuration Table *****
*
*Msg-Block-Type  Byte-Size      Number      Warning-level      Reserve
*-----          -
SMALL            256             50          10                 2
MEDIUM          5000            10          2                  1
```

```

LARGE                64000                3                1                0
%EOS
*
*
* QUEUE CONFIGURATION SECTION
*
* The Queue Configuration Section begins with %QCT, and contains lines
* defining permanent queues. Each line in the QCT section defines a single
* queue. All parameters for a given queue must appear on the same line. The
* general format of a queue definition is given below
*
* QNAME QNUMBER BYTE_QUOTA MSG_QUOTA QUOTA_ENBL UCB_SEND TYPE OWNER CONF_STYLE
* PERM_ACT SCOPE SECURITY
*
* QNAME      - A name for this queue, 255 characters max
*
* QNUMBER    - Number of this queue. Queue numbers must be less than the
*              value established in the %PROFILE section for FIRST_TEMP_QUEUE
*              and may appear only once in the QCT section.
*
* BYTE_QUOTA - The maximum number of uncollected bytes that may reside in this
*              queue. If the value is -1 then the default value is used from
*              queue 0. If there is no queue 0 then default value is 64000.
*
* MSG_QUOTA  - The maximum number of uncollected messages that may reside in
*              this queue. If the value -1 then the default value is used
*              from queue 0. If there is no queue 0 then default value is 100.
*
* QUOTA_ENBL - Controls the enabling/disabling of queue quotas.
*              "All" - all queue quotas on
*              "None" - all queue quotas off
*              "Byte" - only byte quota enabled, msg quota disabled
*              "Msg" - only msg quota enabled, byte quota disabled
*              "." - take the default which is "All"
*
* UCB_SEND   - Send User Callback Indicator (A - J) Indicates which user
*              callback routine is called from PAMS_PUT_MSG to this queue
*
* TYPE       - Type of queue
*              P - primary queue
*              S - secondary queue
*              M - multi-reader queue.
*
* OWNER      - Number of controlling primary queue, if type = "S"
*
* CONF_STYLE - Controls the method by which msgs can be confirmed.
*              "II" - Implicit, in-order confirmations
*              "EI" - Explicit, in-order confirmations
*              "EO" - Explicit, out-of-order confirmations

```

A Sample DMQ\$INIT.TXT File

```

*          "." - take default which is "EO"
*
* PERM_ACT - Permanently active queue: setting this value to "Y" allows
*          processes to insert messages into this queue even if it is
*          not currently attached by any client process.
*
* SCOPE    - "L" for local scope, "G" for global scope
*          globally scoped names are made available to the bus by the
*          name service
*
* SECURITY  - Attach security
*          N - no attach security
*          Y - acl security on attach.  An ACL file must be present
*          and openable for read/write in order to attach the queue
*
*
%QCT          ***** Queue Configuration Table *****
*
*          ---Pool Quota---  UCB  Q  Q  Confirm Perm Name  Chk
*          Queue Name      Num  Bytes Msgs Ctrl Send Type Own  Style Act  Scope ACL
*-----
*   Sample Queues
QUEUE_1      1   64000  100 All      .  .  .  II  Y  L  N
QUEUE_2      2   64000  100 Msg     .  .  .  EI  Y  L  N
QUEUE_3      3   64000  100 Byte   .  .  .  EO  Y  L  N
QUEUE_4      4   64000  100 None   .  .  .  .  Y  L  N
QUEUE_5      5   64000  100 .      .  .  .  .  Y  L  N
QUEUE_6      6   64000  100 .      .  .  .  .  Y  L  N
QUEUE_7      7   64000  100 .      .  .  .  .  Y  L  N
QUEUE_8      8   64000  100 .      .  .  .  .  Y  L  N
QUEUE_9      9   64000  100 .      .  .  .  .  Y  L  N
QUEUE_10    10   64000  100 .      .  .  .  .  Y  L  N
*
*   SBS Server uses the following UCB numbers for Optimized Delivery
*
SBS_ETH_CONTROL  74     0     0  .  E  .  .  .  .  L  N
SBS_ETH_CHAN1   75     0     0  .  E  .  .  .  .  L  N
SBS_ETH_CHAN2   76     0     0  .  E  .  .  .  .  L  N
*
*   Queues 90-100 & 150-199 are reserved for MessageQ utilities
TEMPORARY_Q      0   64000  100  .  .  .  .  .  L  N
SCREEN_PROCESS   0   64000  100  .  .  .  .  .  L  N
SPARE1          90  100000  100  .  .  .  .  Y  L  N
ALL_UCBS        91     0     0  .  .  .  .  .  L  N
TIMER_QUEUE     92     0     0  .  .  .  .  .  L  N
NULL            93     0     0  .  .  .  .  .  L  N
NA_SERVER       94  1000000 1000 None .  .  .  N  L  N
QTRANSFER_SERVER 95  1000000 1000 None .  .  .  N  L  N
DEAD_LETTER_QUEUE 96   64000  100  .  .  .  .  Y  L  N

```


A Sample DMQ\$INIT.TXT File

```
*      *      Transmit SILO  Receive SILO  Maximum      Poll      Dead Poll
*      *      MOT      (in MABs)      (in MABs)      Heartbeat  Interval  Interval
*      REGISTER 5101      1      15      4      10      10
*      REGISTER 5102      1      12      4      10      10
*      REGISTER 5156      1      6      6      10      10
*      *
*END_COMM_SERVICE
*
*
%EOS
*
*
* MRS SECTION
*
* this section is used to specify the journal paths for the recovery,
* post-confirmation and dead-letter journals.
*
%MRS      ***** MRS/JRN Servers Initialization Section *****
*
AREA_SIZE      512 ! disk blks per file (min:128, max:16384, def:512)
NUM_DQF_AREAS 1000 ! min:100, max:1000000, default:1000
NUM_SAF_AREAS 1000 ! min:0, max:1000000, default:1000
NUM_PCJ_AREAS 1000 ! min:0, max:1000000, default:1000
NUM_DLJ_AREAS 1000 ! min:0, max:1000000, default:1000
NUM_MESSAGES  512 ! min:128, max:2147483647, default:512
NUM_QUEUES    128 ! min:128, max:2147483647, default:128
CACHE_PERCENTAGE 90 ! % rcv msg quota MRS msgs (min:1,max:100,def:90)
USE_HIGH_WATER_MARK YES ! checkpt MRS sizing params to disk (YES/NO)
LOAD_MRS_CTRS YES ! init recoverable msg ctrs on startup (YES/NO)
RCVR_ONLY_CONFIRM YES ! limit msg confirms to receiving process (YES/NO)
XGRP_JRN_CTRL NO ! allow JRN cntrl msgs from other groups (YES/NO)
REDELIVERY_TIMER 10 ! integer seconds (min:0, max:5000, default:10)
*
PCJ_FILENAME   DMQ$MRS:MRS_%bg.PCJ ! char[64]- %bg is a macro that
DLJ_FILENAME   DMQ$MRS:MRS_%bg.DLJ ! char[64] - expands to bus_group
*
%EOS
*
* GROUP NAME TABLE SECTION
*
* The Group Name Table begins with %GNT, and contains lines that define
* name-to-queue translations for names and queues that do not necessarily
* appear in the QCT. Each line in the GNT section defines a single
* name-to-queue translation.
*
* NAME  GROUP.QUEUE      SCOPE
*
* NAME      - Case sensitive equivalence name
*
```

```

* GROUP.QUEUE - Group and Queue number to be returned to the calling process
*               when this name is translated via the PAMS_LOCATE_Q service.
*               Specifying 0 for the 'GROUP' implies the local group number.
*
* SCOPE        - Scope of name
*
*               L - Local; the name is loaded into the Group Name Table, but not
*               the distributed naming service. It can be translate by the
*               PAMS_LOCATE_Q service by other processes in this group.
*
*               G - Global; the name is loaded into both the Group Name Table and
*               the distributed naming service. It can be translated by the
*               PAMS_LOCATE_Q service by other processes in this network.
*
%GNT ***** Group Name Table Section *****
*
*           Queue Name                Group.Queue        Scope
*-----
*GLOBAL_QUEUE1                1.234                G
LCL_QUEUE1                    134                  L
LCL_QUEUE2                    135                  L
LCL_QUEUE3                    136                  L
IVP_test_bindq1              0.0                  L
IVP_test_bindq2              0.0                  L
IVP_private_MOT1            4999                  L
IVP_universal_MOT1          5001                  L
*
* alias names for reserved queues
AVAIL_SERVER                  99                    L
DECLARE_SERVER                100                   L
CONNECT_SERVER                100                   L
QUEUE_SERVER                  100                   L
PAMS_TRANSPORT                100                   L
*
%EOS
*
%NAM ***** Naming Agent Section *****
* This section consists of a maximum of 2 entries consisting of
* a keyword, "NA_GROUP", followed by the group number of a group where
* a naming agent is running
*
*NA_GROUP 1
*NA_GROUP 2
*
%EOS
*
%END

```

A *Sample DMQ\$INIT.TXT File*

B Sample DMQ\$TYPCLS.TXT File

Listing B-1 is a sample of the BEA MessageQ DMQ\$TYPCLS.TXT file that is used to define all the message definitions used within the BEA MessageQ system.

Listing B-1 Sample BEA MessageQ DMQ\$TYPCLS.TXT File

```
* Function: *
* This include file defines all the message definitions *
* used within the MessageQ system. *
* *
* NOTE: *
* The following are reserved to DIGITAL: *
* *
* Msg Classes: 28, 29, 62 & 30,000 thru 32,767 *
* Msg Types: -1 thru -5,000 *
*-----*

*****
* *
* MRS Class and Type codes (800-899) *
* *
*****

msg_clas_MRS 28 * define MRS class *

* define MRS type codes *
msg_type_mrs_dqf_transfer -700
msg_type_mrs_dqf_transfer_ack -701
msg_type_mrs_dqf_transfer_rep -702
msg_type_mrs_dqf_set -704
msg_type_mrs_dqf_set_rep -705
msg_type_mrs_saf_set -706
```

B Sample DMQ\$TYPCLS.TXT File

```
msg_type_mrs_saf_set_rep      -707
msg_type_mrs_ack              -801
msg_type_mrs_saf_transfer     -805
msg_type_mrs_saf_transfer_ack -806
msg_type_mrs_saf_transfer_rep -807
msg_type_mrs_cnf              -840
msg_type_mrs_cnf_ack          -841
msg_type_mrs_cnf_nak          -842
msg_type_mrs_debug_on         -850
msg_type_mrs_debug_off        -851
msg_type_mrs_jrn_disable      -871
msg_type_mrs_jrn_disable_rep  -872
msg_type_mrs_jrn_enable       -873
msg_type_mrs_jrn_enable_rep   -874
msg_type_mrs_set_pcj          -880
msg_type_mrs_set_pcj_rep      -881
msg_type_mrs_set_dlj          -882
msg_type_mrs_set_dlj_rep      -883

*****
*
*      PAMS      Class and Type codes (900-999)      *
*
*****

msg_clas_PAMS 29      * define PAMS class *

* define PAMS type codes *
msg_type_timer_expired      -900
msg_type_list_all_q_req     -960
msg_type_list_all_q_resp    -961
msg_type_enable_q_notify_req -962
msg_type_enable_q_notify_resp -963
msg_type_disable_q_notify_req -964
msg_type_disable_q_notify_resp -965
msg_type_q_update           -966
msg_type_locate_q_rep        -972
msg_type_linkmgt_req         -975
msg_type_linkmgt_resp        -976
msg_type_declare_sq          -980
msg_type_undeclare_sq        -981
msg_type_allocated_sq        -982
msg_type_msg_status          -983
msg_type_dmq_server_nak      -989
msg_type_enable_notify       -990
msg_type_disable_notify      -991
msg_type_process_dcl         -992
msg_type_process_exit        -993
msg_type_list_dcls           -994
```

```

msg_type_list_all_entrys          -995
msg_type_list_all_entries        -995
msg_type_list_all_connections    -996
msg_type_list_all_groups        -997
msg_type_link_lost              -998
msg_type_link_complete          -999

*****
*
*      SBS_SERVER - class: PAMS
*                  type: codes (1150-1179)
*
*****

msg_type_sbs_reg                 -1150 * registration message long form
msg_type_sbs_reg_ez             -1173 * registration record short form (new v2
format)
msg_type_sbs_reg_reply         -1152 * registration reply
msg_type_sbs_reg_ez_reply      -1153 * registration reply
msg_type_sbs_dereg_by_id       -1154 * deregister request (old v1 format)
msg_type_mot_dereg             -1154 * deregister request (old v1 format)
msg_type_sbs_dereg            -1174 * deregister request (new v2 format)
msg_type_sbs_dereg_ack         -1155 * deregister acknowledgement
msg_type_sbs_purge            -1157 * purge registration table
msg_type_sbs_purge_ack        -1158 * purge registration acknowledgement
msg_type_sbs_dump_tables      -1162 * process, reg, and group tables to log
msg_type_sbs_bs_seqgap        -1166 * declaration message missing

*****
*
*      AVAIL_SERVER - class: PAMS
*                  type: codes (1180-1189)
*
*****

msg_type_avail_reg             -1180 * available registration
msg_type_avail_dereg          -1181 * available deregistration
msg_type_avail_reg_reply      -1182 * registration reply
msg_type_avail                -1183 * process is now available
msg_type_unavail              -1184 * process is now unavailable

*****
*
*      ETHERNET   Class and Type codes (1000-1010)
*
*****

msg_clas_Ethernet 100      * define Ethernet class

```

B Sample DMQ\$TYPCLS.TXT File

```
* define Ethernet type codes
msg_type_E_connect      -1000 * Initiate connection
msg_type_E_disconnect   -1001 * shutdown circuit
msg_type_E_initialize    -1002 * set Ethernet characteristics
msg_type_E_connect_complete -1003 * circuit established
msg_type_E_connect_rejected -1004 * partner rejected circuit
msg_type_E_partner_disc  -1005 * partner disconnected circuit
msg_type_E_io_error      -1006 * bad return status from QIO$
msg_type_E_protocol_error -1007 * unrecognized msg from partner
msg_type_E_listen_timeout -1008 * partner's heartbeat stopped
msg_type_E_messages_lost -1009 * detected msg loss
msg_type_E_runtime_error -1010 * detected runtime error in AST
```

```
*****
*           Generic UCB Types and Classes           *
*****
```

```
msg_clas_ucb      102
```

```
msg_type_ucb_connect      -1200 * Initiate connection
msg_type_ucb_disconnect   -1201 * shutdown circuit
msg_type_ucb_initialize    -1202 * set UCB characteristics
msg_type_ucb_connect_complete -1203 * circuit established
msg_type_ucb_partner_disc  -1204 * partner disconnected circuit
msg_type_ucb_messages_lost -1205 * detected msg loss
msg_type_ucb_io_error      -1206 * bad return status from QIO$
msg_type_ucb_rcv_data      -1207 * data message received by UCB
```

```
*****
*
*           DEFINE DEMO type codes for DEMO_IO_SERVER           *
*
*****
```

```
msg_type_demo_add      -1200 * add record function *
msg_type_demo_chg      -1216 * chg record function *
msg_type_demo_del      -1232 * del record function *
msg_type_demo_inq      -1248 * inq record function *
msg_type_demo_lst      -1280 * list record function *
msg_type_demo_lst_next -1281 * list next record function *
msg_type_demo_client_up -1282 * client up *
msg_type_demo_client_down -1283 * client down *
```

```
*****
*
*           DEFINE LU62 type codes for LU6.2 UCB           *
*
*****
```

```

msg_clas_appc 62      * define LU6.2 class

msg_type_lu62_init           -6200      * init the interface
msg_type_lu62_define_lu      -6201      * define an LU
msg_type_lu62_allocate       -6202      * allocate a conversation
msg_type_lu62_send_data      -6203      * send a data block
msg_type_lu62_req_confirm    -6204      * ask the remote to CONFIRM
msg_type_lu62_send_confirm   -6205      * reply to a CONFIRM REQ
msg_type_lu62_confirm_rcv    -6206      * Req CONFIRM, go to rcv when obtained
msg_type_lu62_deallocate     -6207      * deallocate a conversation
msg_type_lu62_send_error     -6208      * Signal an error
msg_type_lu62_delete_lu     -6209      * Delete an LU

msg_type_lu62_rcv_data       -6210      * data received from remote partner
msg_type_lu62_confirm_req    -6211      * remote wants a CONFIRM
msg_type_lu62_confirm_send   -6212      * remote has CONFIRMED, gone to rcv
msg_type_lu62_confirmed     -6213      * remote has CONFIRMED
msg_type_lu62_error         -6214      * Error has been signalled
msg_type_lu62_deallocated    -6215      * Conversation has been deallocated

*****
*
* The following message types are for 2.1 functionality
*
*****

msg_type_lu62_req_send       -6220      * Issue REQUEST_TO_SEND
msg_type_lu62_connected     -6221      * Remote has allocated a conversation
msg_type_lu62_define_tp     -6222      * Define TPN for remote to use
msg_type_lu62_activate      -6223      * Wait for inbound connect
msg_type_lu62_ok_send       -6224      * Remote is in RECEIVE State
msg_type_lu62_flush         -6225      * Force transmission of data

*****
*
* The following message types are for the Generic Port Server*
*
*****

msg_type_register_target     -6280      * register a VAX TP
msg_type_connect_request     -6281      * ask for a connection to IBM
msg_type_connect_accept     -6282      * Connection established
msg_type_connect_reject     -6283      * Connection request rejected
msg_type_connection_terminated -6284      * Connection broken
msg_type_change_direction    -6285      * IBM issued RECEIVE
msg_type_data_message        -6286      * Application data / control
msg_type_log_event          -6287      * Event Log Message
msg_type_lu62_add_lu        -6288      * dynamic LU definition

```

B *Sample DMQ\$TYPCLS.TXT File*

```
msg_type_lu62_add_target      -6289  * dynamic Target definition
msg_type_lu62_report_status  -6290  * report LU status to event stream
msg_type_lu62_shutdown      -6299  * DIVE DIVE DIVE

***** End of PAMS_xxx_TYPE_CLASS *****
```

C Parameter Tuning Guidelines

Table C-1 describes the result of increasing or decreasing various OpenVMS and BEA MessageQ parameters.

Table C-1 Parameter Tuning Guidelines

Action	Advantage	Disadvantage	Min/Max Value	Parameter Affected
Increase message buffer count	Allows higher number of pending messages.	MessageQ consumes more page file and global pages, and the page count increases for each process.	8–50,000 for each buffer pool	GBLPAGFIL GBLPAGES VIRTUALPAGECNT
Decrease message buffer count	Frees up page file for other programs.	Increases the probability of lost messages or poor performance due to resource exhaustion.	.	.

C Parameter Tuning Guidelines

Increase message buffer reserve count	Allocates a higher number of message buffers for exclusive MessageQ internal use. Results in higher probability of unblocking and internal messages being delivered successfully.	Reduces the available number of message buffers for general use.	0 to all buffers in pool	.
Decrease message buffer reserve count	Increases available number of buffers.	Increases probability of lost messages or poor performance due to resource exhaustion.	.	.
Increase message buffer size	Allows transfer of larger blocks of data in a single call.	Consumes extra memory when data fills only a small portion of the buffer. Increases the size of each MessageQ process due to the creation of the local MessageQ buffers; two to four buffers depending on the features utilized.	1–31998 (small) 2–31999 (medium) 1500–65000 (large) S < M < L	PGFLQUO GBLPAGFIL GBLPAGES VIRTUALPAGECNT

Decrease message buffer size	Higher utilization of available memory.	Increases program complexity to disassemble and reassemble the data buffers when larger data transfers are required. Some features of MessageQ might not work because of the inability to allocate a large enough buffer.	.	.
Increase receive message quota	Allows a higher number of pending messages to build up during a burst condition.	A queue might consume a disproportionate amount of resources on its pending queue, causing other parts of the system to have problems.	0 bytes– 2 GB	.
Decrease receive message quota	Limits resource consumption of slow service rate queues or queues with no reader.	Queues might be unable to receive large messages. Messages might be lost or delayed because they cannot be queued.	.	.
Increase MRS NUM_DQF_AREAS	Allows more unconfirmed messages to be stored before a UMA is taken.	Uses more disk space and MRS Server resources.	100– 1000000	.
Decrease MRS NUM_DQF_AREAS	Reduces disk and MRS Server resource consumption. Reduces time to locate a given message.	Reduces probability of storing a message.	.	.

C Parameter Tuning Guidelines

Increase MRS CACHE_PERCENTAGE	Allows the queue to have more outstanding messages, which results in more overlap of the MRS Server I/O when applications process recoverable messages. This quota is a percentage of the byte or message receive quota.	Could cause the MRS Server to consume most or all of the receive byte/message quota on queue attachment.	1–100	.
Decrease MRS CACHE_PERCENTAGE	Reduces consumption of MessageQ buffers on queue attachment.	Lowers the amount of I/O overlap for reading messages out of the DQF and delivering the messages to the appropriate queue.	.	.
Increase MRS Maximum Queues	Allows the MRS Server to track more local and remote queues.	Causes the MRS Server to grow larger to contain these data structures.	128– 2147483647	FILLM PGFLQUO VIRTUALPAGECNT
Decrease MRS Maximum Queues	Reduces the size of the MRS Server.	The MRS Server might be unable to track all the queues requested.	.	.
Increase MRS Maximum Messages	Allows the MRS Server to track more messages	Causes the MRS Server to grow larger to contain these data structures.	128– 2147483647	PGFLQUO VIRTUALPAGECNT
Decrease MRS Maximum Messages	Reduces the size of the MRS Server.	The MRS Server might be unable to track all the messages requested.	.	.

Increase MRS NUM_SAF_AREAS	Allows the MRS Server to handle more remote queues.	Could increase the number of files and disk blocks used by the MRS Server.	0-1000000	.
Increase MRS NUM_SAF_AREAS	Limits the total number of remote queues that can have pending SAF messages.	Might be unable to track all queues requested.	.	.
Increase MRS AREA_SIZE	Reduces disk fragmentation and number of files needed to store messages.	Less efficient use of disk space.	128-16384	FILLM blocks for each file.
Decrease MRS AREA_SIZE	More efficient use of disk space.	Could cause an inordinate number of disk files to be created. Causes disk and directory fragmentation.	.	.
Increase MRS Redelivery Timer	Less CPU overhead due to reduced retries.	Less responsive to changes in available resources.	0-5000	.
Decrease MRS Redelivery Timer	More responsive in systems with limited resources.	Could cause an inordinate amount of CPU attempting to retry message delivery.	.	.

D Directories and Logical Names

This appendix lists the BEA MessageQ directories and logical names.

Directories

Table D-1 lists the BEA MessageQ directories.

Table D-1 BEA MessageQ Directories

Directory	Logical	Description
[DMQ\$Vmm]	.	BEA MessageQ root area.
[.CVT]	.	BEA MessageQ Conversion Utility
[.DMQNS]	.	Default lightweight namespace area.
[.DOC]	DMQ\$DOC	Documentation files and release notes.
[.EXAMPLES]	DMQ\$EXAMPLES	Sample programs.
[.EXE]	DMQ\$EXE	Executable files and command files.
[.LIB]	DMQ\$LIB	Library holding area for standard object and text libraries.
[.LOCAL.C]	.	Catalog files for localization.
[.MRS . bbbb_ggggg]	DMQ\$MRS	MRS Server data file area for DQFs and SAFs.

D Directories and Logical Names

[.USER . bbbb_ggggg]	DMQ\$USER DMQ\$ACCESS	Specific user customizing area. The extension is the 4-digit bus ID and 5-digit group ID.
[.LOG . bbbb_ggggg]	DMQ\$LOG	Specific group server log file area.
[.UDATAOBJ]		Directory where the BEA MessageQ license file is to be installed (MSGQnn_LIC.TXT)
[.USER . CLIENT]		Initialization and header files for the BEA MessageQ Client for OpenVMS.
[.USER . TEMPLATE]		User customizing area template. Files should be moved before attempting to alter any files.

Logical Names

Table D-2 lists the BEA MessageQ logical names. The second column in the table refers to whether the logical names are dynamically translated when a program attaches its first queue.

Table D-2 BEA MessageQ Logical Names

Logical Name	Dynamic	Description
BEADIR	.	Directory specification for BEA MessageQ root area. This logical name is used by the message catalogs.
DMQ\$ACCESS ¹	.	Directory specification of BEA MessageQ access control files.
DMQ\$BLOCKING_CONFIRM ¹	.	Controls whether or not the PAMS_CONFIRM_MSG routine waits for a response from the recovery system.
DMQ\$CHKPT_FILE ¹	.	File specification of a file that counts the number of times the COM Server has been booted. Generates unique sequence numbers.
DMQ\$BUS_GROUP ¹	Y	Contains the current bus ID and group ID.

DMQ\$DCL_NUMBER ³	Y	Base ten number that overrides the queue number submitted to PAMS_ATTACH_Q.
DMQ\$DEBUG ³	Y	Controls the built-in BEA MessageQ debug code: <ul style="list-style-type: none"> ■ ERROR = Display only error messages. ■ TRACE = Trace module calls. ■ ALL = Display both trace and error messages.
DMQ\$DEBUG_MEMORY ³	Y	Enables tracing of global memory structures. For troubleshooting purposes only.
DMQ\$DEBUG_SELECT ³	Y	Enables trace dumping of the selection list. For troubleshooting purposes only.
DMQ\$DEF_XGRP_BUF_POOL	Y	Overrides the default cross group buffer value on an unsolicited connection.
DMQ\$DEFAULT_SEND_TMO ³	Y	Overrides the timeout value for the pams_put_msg service.
DMQ\$DISABLE_FORCEX ³	Y	Controls whether the COM Server forces all processes that are attached to the COM Server to exit along with it. The default is NO. YES = COM Server does not perform the forced exit.
DMQ\$DISABLE_SYSOUT ³	Y	Controls whether BEA MessageQ logs events to SYS\$OUTPUT. YES = Printing disabled.
DMQ\$DISK ¹	.	Device specification of the disk where BEA MessageQ is installed.
DMQ\$DOC ¹	.	Directory specification of BEA MessageQ documentation.
DMQ\$DNS_DIRECTORY ³	.	Default DNS namespace and specification used by various BEA MessageQ routines.
DMQ\$ENTRYRTL ¹	.	File specification of the installed BEA MessageQ run-time library DMQ\$ENTRYRTL n .EXE.
DMQ\$ERROUT ²	Y	Device specification of the alternate BEA MessageQ console device. If not specified, BEA MessageQ messages are printed on the system console.

D Directories and Logical Names

DMQ\$EXAMPLES ¹	.	Directory specification of BEA MessageQ example programs.
DMQ\$EXE ¹	.	Directory specification of BEA MessageQ executable modules.
DMQ\$EXECRTL ¹	.	File specification of the installed EXEC mode BEA MessageQ routines in DMQ\$EXECRTLVnn.EXE.
DMQ\$EXIT_PURGE ³	Y	Controls the purging of pending LLS messages when the process exits.NO = do not purge messages.
DMQ\$GMT_OFFSET ³	Y	The time difference in hours between local system time and Greenwich Mean Time (GMT). The value of this logical name must be set for TCP/IP network communications.
DMQ\$GROUP_SYNCHRONIZE ⁴	.	Enables automatic synchronized cluster group failover. Setting this to NO will require that failover be managed by the user. The default is YES.
DMQ\$HEADERS ³	Y	Controls the printing of BEA MessageQ message headers on the user's terminal. YES = print message headers.
DMQ\$INIT_FILE ₁	.	File specification of the BEA MessageQ initialization file.
DMQ\$LIB ₁	.	Directory specification of the BEA MessageQ libraries.
DMQ\$LIBGP ¹	.	File specification of the BEA MessageQ shareable image DMQ\$LIBGPVnn.EXE
DMQ\$LIBFML32 ¹	.	File specification of the BEA MessageQ shareable image DMQ\$LIBFML32Vnn.EXE
DMQ\$LIC_FILE ¹	.	File specification of the BEA MessageQ license file.
DMQ\$LOG ¹	.	Directory specification of the BEA MessageQ log area.
DMQ\$MRS ₁	.	Directory specification for the MRS data files.
DMQ\$MRS_CHECK_ML ⁵	.	Enables MRS validation scan of the internal MRS message list. For troubleshooting purposes only.

DMQ\$MRS_CHECK_PL ⁵	.	Enables MRS validation scan of the internal MRS process list. For troubleshooting purposes only.
DMQ\$MRS_CHUNK_SIZE ⁴	.	Controls the journaling chunk size for DQF and SAF journals.
DMQ\$MRS_CLEANUP_ON_CLOSE ⁵	.	Enables MRS to delete a DQF or SAF journal file on close if the file is empty.
DMQ\$MRS_DISABLE_JOURNALING ⁴	.	Disables all journaling. This logical is processed by the COM Server only. Default is NO.
DMQ\$MRS_DISABLE_SH ⁴	.	<p>Disables submitter history checking when set to YES. The default is NO. To use this logical name, define it in DMQ\$SET_SERVER_LOGICALS.COM in the MRS section.</p> <p>Disabling submitter history checking increases the chances of duplicate recoverable SAF messages being delivered to the receiving application. Applications must be capable of handling duplicate messages before enabling this logical. Duplicate SAF messages may occur in non-FIFO order, typically with a single message stutter caused by a link transition.</p>
DMQ\$MRS_DQF_CONTROL ⁵	.	Disables operation of DQF files unless explicitly requested by a control message.
DMQ\$MRS_FREE_FILE_MAX ⁴	.	Limits the number of FREE files that the MRS Server may create during recoverable messaging. The default value is calculated by the MRS Server based on the NUM_DQF_AREAS and NUM_SAF_AREAS (DMQ\$INIT.TXT). Cannot be smaller than DMQ\$MRS_FREE_FILE_MIN.
DMQ\$MRS_FREE_FILE_MIN ⁴	.	Controls the minimum number of FREE files that the MRS Server may use during recoverable messaging. The default value is 10. To turn the Free File Engine OFF, set it to 0.
DMQ\$MRS_READ_AHEAD_DEPTH ⁵	.	Controls the read ahead buffer depth. This value can be tuned to increase MRS read I/O performance. The default value is 4. It is not recommended that this value be changed.

D Directories and Logical Names

DMQ\$MRS_RING_DEPTH ⁵	.	Controls the depth of the confirmation ring buffer. The default value is 10.
DMQ\$MRS_SAF_CONTROL ⁵	.	Disables operation of SAF files unless explicitly requested by a control message.
DMQ\$MRS_SERVER_CHECK_TIMER ⁵	.	Internal time which MRS uses to check the state of adjacent MRS servers. Value is in tenths of seconds with a valid range of 30 to 3000 seconds (5 minutes). Default is 600 (1 minute).
DMQ\$MRS_TRACE_PL ⁵	.	Enables MRS tracing of the internal MRS process list. For troubleshooting purposes only.
DMQ\$MRS_TRACE_STATE ⁵	.	Enables MRS tracing of internal MRS state changes. For troubleshooting purposes only.
DMQ\$MSGSHR ¹	.	File specification of the BEA MessageQ message table.
DMQ\$OBJECT_CACHE_SIZE ³	Y	Sets the maximum number of queue names that be stored the PAMS_LOCATE_Q process cache. Valid setting are 20 to 32000. The default value is 128.
DMQ\$POOL_CHK_WARNING_INTERVAL ⁴	.	Controls the number of seconds between low buffer pool warning messages. The default is 120 seconds.
DMQ\$PSSRTL ¹	.	Supplied for backwards compatibility. Points to DMQ\$ENTRYRTL.
DMQ\$OPCOM_TARGET ²	Y	Redirects console event logging to OPCOM. See Chapter 12, “Managing a BEA MessageQ Environment,” for a full description.
DMQ\$GROUP_OUTPUT ³	Y	Sets output direction to any of the following file output streams: CONSOLE, EVL_LOG, USER_LOG, SYSOUT, SYSERR, MEMLOG, or NONE.
DMQ\$PROCESS_OUTPUT ³	Y	Sets output direction to any of the following file output streams: CONSOLE, EVL_LOG, USER_LOG, SYSOUT, SYSERR, MEMLOG, or NONE.
DMQ\$TRACE_OUTPUT ³	Y	Sets output direction to any of the following file output streams: CONSOLE, EVL_LOG, USER_LOG, SYSOUT, SYSERR, MEMLOG, or NONE.

DMQ\$QTRANSFER_MAX_PROCS ⁵	.	Defines the number of concurrent streams the Qtransfer Server can handle at one time. The default is 5.
DMQ\$ROOT ¹	.	The root directory name of the BEA MessageQ software being used (for example, DMQVnn).
DMQ\$SBS_ETH_MOT_ONLY ⁵	.	Allows only Ethernet broadcasting via the SBS server when set to YES. The default is NO.
DMQ\$SBS_MAX_ASM_Q ⁵	.	Overrides the default number of SBS assembly queues. The default is 10.
DMQ\$SBS_MAX_MSG_SIZE ⁵	.	Overrides the default size of the SBS assembly queue. Can not exceed 32000.
DMQ\$SBSE_REXMIT_TIMEOUT ⁵	.	Overrides the default retransmission timer. The default is 30 seconds.
DMQ\$SBSE_mmmm_MAX_HB ⁵	.	Overrides the maximum heartbeat interval for the specified MOT mmmm defined in the group initialization file.
DMQ\$SBSE_mmmm_POLL_INT ⁵	.	Overrides the polling interval for the specified MOT mmmm defined in the group initialization file.
DMQ\$SCRIPT ³	Y	Controls script processing. This logical should provide the file specification for the script file. If the script file cannot be found then the user will be prompted for one. A value of N will disable scripts.
DMQ\$SYMBOLS ³	Y	Specifies the type-class file to be loaded for script processing. The default is DMQ\$USER:P_TYPECL.H.
DMQ\$SET_GBLSEC_PROT ²	.	Overrides the protection mask on the BEA MessageQ global sections.
DMQ\$SET_LNM ¹	.	File specification for the installed BEA MessageQ shareable image DMQ\$SET_LNM_TABLEVnn.EXE.
DMQ\$SERVER_TRACE ³	.	Controls the tracing of BEA MessageQ Server-specific information. YES = Trace on. This logical generates many trace lines for each message in the server log file.
DMQ\$SPAWN_SERVERS ²	.	Suppresses the start of BEA MessageQ servers when it is set to NO.

D Directories and Logical Names

DMQ\$VERIFY_ON ³	Y	Enables verification of global sections in the DMQ\$LLS_VERIFY utility. The default is NO.
DMQ\$PRIMARY_DECNET_LD ²	.	Controls which DECnet link driver is the primary driver. The default is v2. Setting this logical to v3 will force the DECNETV3 driver to be the primary DECnet link driver and the DECNETV2 (COM Server) driver to be the secondary DECnet link driver.
DMQ\$TCP_IP_LD ¹	.	Controls which TCP/IP link driver is used.
DMQ\$TCP_IP_CON_ABORT ⁵ DMQ\$DECNET_CON_ABORT ⁵	.	Controls the connection abort timer. When a link connection is being established, at the beginning of the protocol handshake, the connect abort timer is started. The timer is canceled when the link is successfully established. The default value is 90 seconds.
DMQ\$TCP_IP_LD_NAME ⁵	.	Override the default hostname returned by a call to <code>gethostname()</code> . This logical name can be used to define an alternate interface for the linkdriver activity.
DMQ\$USER ¹	.	Directory specification for the BEA MessageQ user area.
DMQ\$VPS_USER ¹	.	File spec of the BEA MessageQ shareable image DMQ\$VPS_USERVnn.EXE.
DMQ\$VPS_EXEC ¹	.	File spec of the BEA MessageQ installed shareable image DMQ\$VPS_EXEcvnn.EXE.
DMQLD_TRACE_LEVEL ⁵	.	Defines and overrides the value for LD tracing. This value is a bit mask which is defined as follows: <ul style="list-style-type: none">■ 1 = skinny trace, I/O completions, message processing■ 2 = dump data from each I/O■ 4 = full tracing of all routines
DMQLD_PING_TIMEOUT ⁵	.	Defines how long, in seconds, we wait to hear from a V4 group as a result of a ping request. The default is set to 90 seconds.
DMQLD_PING_INTERVAL ⁵	.	Defines the interval between pings. The default value is 30 seconds.

DMQLD_MAX_DATAIO_LEN ⁵	.	Defines the maximum number of bytes that will be used on a single write to the link. The default value is 65534 bytes.
DMQLD_TCP_UNREACH_RETRY ⁵	.	In the case where a TCP stack reports a warning for NETUNREACH or HOSTUNREACH, a retry is offered for the I/O operation. The retry value can be from 1 to 60 attempts. The default value is 0, do not retry. Note: BEA Systems recommends this retry feature be avoided and the customer focus placed on the repair of the network that is generating these up/down transitions between hosts.
DMQLD_TCP_UNREACH_DELAY ⁵	.	Used in conjunction with the DMQLD_TCP_UNREACH_RETRY to vary the delay between retry attempts. This value is not used if DMQLD_TCP_UNREACH_RETRY is not enabled. The default value is 60 seconds.
DMQNS_DEFAULTPATH	.	Directory specification of the namespace root directory.
DMQNS_DEVICE	.	The default pathname for the namespace (based on the definition for DMQNS_DEVICE).

¹Set by DMQ\$BOOT.COM.

²Optionally set by DMQ\$BOOT.COM.

³Optionally set by user at DCL level.

⁴Set by DMQ\$SET_SERVER_LOGICALS.COM.

⁵Optionally set by DMQ\$SET_SERVER_LOGICALS.COM.

E Error Log Messages

This appendix provides the following error log information:

- COM Server and Link Driver error log messages
- MRS and Journal Server error log messages
- SBS Server error log messages

COM Server and Link Driver Error Log Messages

This section lists and describes COM Server and Link driver error log messages. Use the following Key to understand conventions used in this section:

Key:

- *g* = Group ID decimal value
- *q* = Queue number decimal value
- *n* = A numeric value
- *x* = Hexadecimal value
- *s* = Character string value
- *Italic* text represents information that can vary in log messages.

Accepting connect from system *system_name* (as alias *alias_name n.n.n.n*) for group *g* (*group_name*)

Explanation: Informational.

A connect request from a system identifying itself as *system_name* has been received, but that system name does not exist in the MessageQ initialization file and/or the local network database. However, a different system name (*alias_name*) was located and it uses the same network address as *system_name*. The Link Driver assumes that one name is an alias for the other, and that they both refer to the same system.

The message indicates that the connection will be accepted, but under the name *alias_name* instead; to match existing definitions in the MessageQ initialization file and enforce consistency. All further console messages referring to this connection will display the *alias_name* system name, as will the MessageQ Monitor program.

User Action: If aliases are being used and both names refer to the same system, no action is necessary.

If the indicated system names *system_name* and *alias_name* do not refer to the same system, this indicates there are two systems on your network with the same network address, or with inconsistent network databases. You'll need to correct this situation and update the MessageQ initialization file if necessary.

Accepting connect from system *system_name* for group *g* (*group_name*)

Explanation: Informational.

The server is accepting a network connection from another server with the indicated group ID. This message appears when a group connection requires two data links, and indicates that the inbound link is up.

User Action: None.

Accepted connect from system *system_name* for group *g* (*group_name*)

Explanation: Informational.

The server has accepted a network connection from another server with the indicated group ID, and the group connection is now available for use.

User Action: None.

An error occurred while freeing messages for process *q* PID (*pid*)

Explanation: Error.

The server, while attempting to clean up a process's queues, detected an error.

User Action: Contact MessageQ support with a description of the problem.

Attempt to double define group *g* into the RT

Explanation: Warning.

While loading a Routing Table entry, MessageQ found the same group definition in the Cross-Group Connection Table. Group entries can be created by the MessageQ Loader utility or by an unsolicited connection.

User Action: Determine the preferred style of connection to the group and remove the other entry.

Bad DECnet connect IOSB status for link to group *g* (*group_name*) on system *system_name*

Explanation: Warning.

An error was returned by DECnet when attempting to connect to another servers group.

User Action: Examine the error and determine if the cause of the error is due to a local DECnet error, remote system error, or a configuration error.

Bad PAMS_SET_TIMER return status for X-Group auto reconnect

Explanation: Warning.

MessageQ detected an error while attempting to set a wake-up timer for automatic cross-group reconnection.

User Action: Examine the failure status posted on the next line to determine if failure is due to a process quota problem.

Bad PAMS_SET_TIMER return status for X-group connect

Explanation: Warning.

MessageQ detected an error while attempting to set a timer for canceling the current cross-group reconnection cycle.

User Action: Examine the failure status posted on the next line to determine if failure is due to a process quota problem.

Bad PAMS_SET_TIMER return status for X-group connect to group *g*

Explanation: Warning.

MessageQ detected an error while attempting to set a wake-up timer for a specific cross-group reconnect attempt.

User Action: Examine the failure status posted on the next line to determine if failure is due to a process quota problem.

Bad PAMS_SET_TIMER return status for X-Group link protocol

Explanation: Error.

An error occurred when trying to declare an internal protocol timer. If recoverable, the link driver will attempt to continue.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Bad status from LIB\$ASN_WITH_MBX in XDECNET

Explanation: Warning.

MessageQ detected an error while attempting to either initiate or complete a DECnet connection.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Bad XDECNET_POST_OOB_READ call

Explanation: Warning.

MessageQ detected an error while attempting to post a read \$QIO to a DECnet Mailbox channel.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Bad XDECNET_POST_OOB_READ call for self

Explanation: Warning.

MessageQ detected an error while attempting to post a read \$QIO to the main DECnet Mailbox channel.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Changing outbound for group *g* (*group_name*) to match inbound

Explanation: Warning.

An incoming connection request was received while an outbound connect was being performed to a different system or transport. The current outbound attempt will be dropped in favor of completing the incoming connection request.

User Action: None.

Connection for group *g* (*group_name*) to system *system_name* is down

Explanation: Warning.

All links to the indicated group are down.

User Action: Examine the previous messages for the indicated group and system for more information concerning the action.

Could not locate transport address (port number)

Explanation: Error.

The TCP/IP port number was omitted from the MessageQ initialization file definition for the local group.

User Action: The file must be corrected, or the transport will not be available for use.

DECnet communications lost to message queue group *g* (*group_name*)

Explanation: Warning.

The DECnet link to a remote COM Server aborted.

User Action: None.

DECnet communications lost to message queue group *g* (*group_name*) --- file not accessible

Explanation: Warning.

The current DECnet link to a remote COM Server is inaccessible.

User Action: None.

DECnet communications lost, 3rd party, to message queue group *g* (*group_name*)

Explanation: Warning.

An NCP command to disconnect the link was issued.

User Action: None.

DECnet data overrun from message queue group *g* (*group_name*)

Explanation: Warning.

The incoming message was larger than the internal buffer.

User Action: Compare the large buffer sizes of the two groups and adjust.

DECnet link exit to message queue group *g* (*group_name*)

Explanation: Warning.

A DECnet link exit was detected.

User Action: None.

DECnet path lost to message queue group *g* (*group_name*)

Explanation: Warning.

The DECnet path was lost to the remote group. This condition might be caused by a failure of the communication hardware.

User Action: Repair as necessary.

Detected DECnet network shutdown

Explanation: Warning.

A DECnet network shutdown is causing the MessageQ network object to be deleted.

User Action: None. The COM Server will automatically recreate the DECnet object soon after the network is restarted.

Dropped link to group g (*group_name*)

Explanation: Warning.

A DECnet link event caused a link to a group to be dropped.

User Action: Examine the failure status on the next line to determine if it is a quota or configuration problem.

Dropped link to group g (*group_name*) due to LLS buffer pool exhaustion

Explanation: Warning.

When attempting to queue a message received from another group, MessageQ found the buffer pool empty.

User Action: Increase the number of buffers in the buffer pool, decrease the receive byte quota for the queues, or change to a delivery mode that will pace the data across the link.

Dropped link to group g (*group_name*)

Explanation: Warning.

The link to the indicated group is being dropped due to an unusable condition.

User Action: Examine the previous messages for the indicated group for more information concerning the action.

Dropped link to remote logical group g which is incompatible with expected group g (*group_name*)

Explanation: Error.

While connecting to another group, the group was configured with a different group ID than what was stored in its own table.

User Action: Change the entries in the Cross-Group Connection Table in the group initialization file as necessary.

Duplicate process notification request from $g.g$ discarded

Explanation: Warning.

An existing notification request was received.

User Action: None.

Duplicate queue notification request from $g.g$

Explanation: Warning.

An existing notification request was received.

User Action: None.

Duplicate Queue number passed to ENQ_PROCESS_LOCK_ASK routine = q

Explanation: Error.

Process Cleanup list has duplicate entries.

User Action: Contact MessageQ support for more detail on this error.

ENQ failed for "lock_name"

Explanation: Warning.

While attempting to create a MessageQ process lock to assist in queue cleanup, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

ENQ failed for DMQCS_GRP_LOCK of group *g*

Explanation: Error.

While attempting to obtain a MessageQ group lock, an error was detected.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

ENQ for "lock_name" not previously locked by process, releasing lock

Explanation: Warning.

While performing a PAMS_ATTACH_Q, a process exited before the COM Server could complete the attachment.

User Action: This problem could be related to processor load, process priorities, or the COM Server logging too many events.

ERROR --- No available temporary queues

Explanation: Error.

When attempting to allocate a temporary queue, MessageQ found the pool of available queues empty.

User Action: Increase the number of temporary queues by lowering the FIRST_TEMP_QUEUE value defined in the group initialization file.

Error converting protocol for group *g* (group_name)

Explanation: Error.

An unexpected error occurred while performing internal protocol conversions. The link to the group will be dropped.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Error during QIO xgroup send - grp:*g*

Explanation: Error.

The indicated QIO to the group shown failed to complete with an unexpected error. The connection to the group will be dropped.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Error while posting X-group rcv QIO for group g - channel # n

Explanation: Error.

The attempt to post the indicated QIO to the group and channel shown failed with an unexpected error. The connection to the group will be dropped.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Error from X410_POST_READ_OOB_CHAN QIO for group g --- channel # n

Explanation: Warning.

While attempting to post a DECnet \$QIO read operation to a link's mailbox channel, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Error returned from X200_POST_CONNECT_REQUEST QIO

Explanation: Warning.

MessageQ detected an error while attempting to connect to a remote group.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Error while posting X-group rcv QIO for group g --- channel # n Dropped connection to group g

Explanation: Warning.

While attempting to post a DECnet \$QIO read operation, MessageQ detected an error.

User Action: Examine the accompanying error status and determine if it is due to the inaccessibility of the remote group.

Exceeded max number of outstanding queue cleanup requests - request q dropped.

Explanation: Warning.

Exceeded the maximum number of outstanding process cleanup requests.

User Action: Reduce the number of processes requiring cleanup.

Exceeded process notification table --- request from g,q discarded

Explanation: Warning.

The number of requesters of process notification messages exceeds the number in the internal table.

User Action: Reduce the number of requesters.

Exceeded queue notification table --- request from g,q

Explanation: Warning.

The number of requesters of queue notification messages exceeds the number

in the internal table.

User Action: Reduce the number of requesters.

Exceeded routing visit count - UMA taken

+ discarded msg --- Src=*g.q* Target=*g.q* class=*n* type=*n*

Explanation: Error.

MessageQ could not deliver the message to its destination queue because the routing visit count has been exceeded. The UMA was taken.

User Action: Recheck routing table and correct as necessary.

Exceeded write quota on link

+ discarded msg --- Src=*g.q* Target=*g.q* class=*n* type=*n*

Explanation: Error.

MessageQ could not deliver the message because the group buffer pool has been exceeded.

User Action: Increase the xgroup buffer pool in the X-Group section of the group initialization file.

Exceeded X-Group notification table --- request from *g.q* discarded

Explanation: Warning.

The number of requesters of cross-group notification messages exceeds the number in the internal table.

User Action: Reduce the number of requesters.

Failed on DMQCS_XDN_GET_READ_BUFFER

Explanation: Fatal.

Failed when attempting to read xgroup DECnet buffer.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed on \$CRELNM of termination mailbox lnm

Explanation: Error.

An error occurred while attempting to create the termination mailbox logical name `DMQ$TERMINATION_MBX`.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed on \$GETDVIW of temporary mailbox for Server spawning

Explanation: Error.

An error occurred while attempting to access mailbox channel.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed on LIB\$GET_VM_PAGE for DECnet MBX

Explanation: Fatal.

When attempting to acquire a DECnet mailbox buffer, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed on LIB\$GET_VM_PAGE for DECnet abort link driver

Explanation: Error.

When attempting to acquire a transport control block buffer, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed on LIB\$GET_VM_PAGE for DECnet TCB buffer

Explanation: Fatal.

When attempting to acquire a DECnet transport control block buffer, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed on LIB\$GET_VM_PAGE for DECnet LCB

Explanation: Fatal.

When attempting to acquire a DECnet LCB buffer, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed on LIB\$SPAWN of Server server_name

Explanation: Error.

An error occurred while attempting to spawn a server process.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed on SYS\$GETJPIW call for DMQ queue *q.q*

Explanation: Warning.

MessageQ detected an error while attempting to perform a SYS\$GETJPIW service. This message is in conjunction with validating a requester's privileges.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed on searching rights list for "username" for DMQ queue *g.g*

Explanation: Warning.

MessageQ detected an error while attempting to validate a privileged request to the COM Server.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed on termination mbx SYSS\$QIO

Explanation: Error.

An error occurred while attempting to post a mailbox read.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed to accept connect request from system *system_name* for group *g*

Explanation: Warning.

MessageQ detected an error while attempting to accept a connection from another group.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or if it is related to processor load.

Failed to accept connect request from system *system_name* for group *g* (*group_name*)

Explanation: Warning.

MessageQ detected an error while attempting to accept a connection from another group. Asterisks (****) for the group name indicate that the group was not previously loaded in the Cross-Group Connection Table.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or if it is related to processor load.

Failed to allocate GROUP entry during connect from group *g* --- Link dropped

Explanation: Warning.

While processing an incoming connect request for a previously unknown group, MessageQ detected an error during the allocation of an entry in the Cross-Group Connection Table.

User Action: Determine whether you need to increase the size of the Cross-Group Connection Table to accommodate the number of groups on the bus, or increase memory quotas.

Failed to allocate local buffer for I/O to group *g* (*group_name*)

Explanation: Error.

The link driver was unable to allocate buffer memory for performing I/O to the indicated group. The link to the group will be dropped.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Failed to allocate group entry during connect from group *g* - Link dropped to system *system_name*

Explanation: Error.

MessageQ failed to allocate a group entry structure.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed to allocate msg buffer for sending X-Group msg to group *g* (*group_name*)

Explanation: Error.

MessageQ failed to allocate a message buffer.

User Action: Examine the failure status posted on the next line to determine if it is due to the MessageQ buffer pool exhaustion.

Failed to allocate msg buffer for group notify msg.

Explanation: Warning.

While attempting to deliver a group notify message, MessageQ failed to allocate a message buffer.

User Action: Examine the failure status posted on the next line to determine if it is due to the MessageQ buffer pool exhaustion.

Failed to allocate msg buffer for notify msg

Explanation: Warning.

While attempting to deliver a process notify message, MessageQ detected an error in the message allocation routine.

User Action: Examine the failure status posted on the next line to determine if it is due to the MessageQ buffer pool exhaustion.

Failed to connect group *g* (*group_name*) to system *system_name* - System name is unknown

Explanation: Error.

A connection attempt could not be made to the indicated group because the system name does not exist in the applicable network database.

User Action: Change the MessageQ initialization file, or add the entry to the network database to correct the condition.

Failed to convert outgoing message

+ discarded msg --- Src=*g.g* Target=*g.g* class=*n* type=*n*

Explanation: Error.

MessageQ could not convert the outgoing xgroup message. The specified

delivery mode and UMA combination are not support by the receiving group.

User Action: Correct as necessary.

Failed to create temporary mailbox for Server spawning

Explanation: Error.

An error occurred while attempting to create the termination mailbox required for server spawning.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed to create termination mailbox read buffer

Explanation: Error.

An error occurred while attempting allocate space for the termination mailbox read buffer.

User Action: Examine the failure status on the next line to determine the cause of the failure.

Failed to map global section --- program exiting

Explanation: Fatal.

The link driver could not start due to being unable to connect to the internal global memory sections.

User Action: Contact MessageQ support with a description of the error.

Failed to create MessageQ DECnet object --- DECnet DISABLED

Explanation: Error.

While attempting to create the MessageQ DECnet named object, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem, DECnet failure, or system resource problem.

Failed to create n DECnet x-group queue heads n pagelets.

Explanation: Fatal.

When attempting to create the DECnet cross-group write list heads, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed to create DECnet x-group write tokens

Explanation: Fatal.

When attempting to create cross-group write tokens, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed to create DECnet x-group write tokens free list

Explanation: Fatal.

When attempting to create cross-group write tokens free list, MessageQ detected an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

Failed to create Event Logger MBX

Explanation: Error.

A failure occurred when attempting to create the Event Logger mailbox.

User action: Examine the failure status posted to determine why the mailbox could not be created.

Failed to create n entry RT Notify Table

Explanation: Error.

MessageQ detected an error while attempting to create the Routing Table data structure.

User Action: Examine the failure status on the next line to determine if it is a quota or configuration problem.

Failed to create n entry Queue Notify table

Explanation: Error.

MessageQ detected an error while attempting to create the Queue Notify Table data structure.

User Action: Examine the failure status on the next line to determine if it is a quota or configuration problem.

Failed to find group entry during connect from group g

Explanation: Warning.

An attempt to locate or add a new entry to the Cross-Group Connection Table resulted in an error.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

**** Failed to init the print stream ****

Explanation: Fatal.

An error occurred while MessageQ attempted to initialize the output streams.

User Action: Examine the failure status on the next line to determine if it is a quota or configuration problem.

Failed to initialize DECnet communication

Explanation: Error.

An error occurred while MessageQ attempted to initialize DECnet communication.

User Action: Examine the failure status on the next line to determine if it is a configuration problem.

Failed to load NT into TCB for group *g* (group_name), table entry = group_index, NT = cur_nt

Explanation: Error.

MessageQ detected an error while attempting to load the NT structure into the Transport Control Block.

User Action: Examine the failure status posted on the next line to determine the source of the problem.

Failed to send group notify msg to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to queue a notification message.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to send notify msg to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to queue a notification message.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to send PAMS request CHKPT_FILE_RESP msg to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to queue a CHKPT_FILE_RESP messages.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to send PAMS request NAK msg to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to send an internal server NAK message. notification message.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to send queue notify msg to g.q

Explanation: Warning.

MessageQ detected an error while attempting to queue a notification message.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to send RT update notify msg to g.q

Explanation: Warning.

MessageQ detected an error while attempting to queue a notification message.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to start network listener

Explanation: Informational.

The link driver could not start because the transport listener failed (most probably because the network is not running).

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Failed to startup the MessageQ Event Logger - Exiting

Explanation: Fatal.

The attempt to start the Event Logger failed.

User Action: Examine the failure status on the next line to determine if it is a quota or configuration problem.

Failed to startup the *server_name*

Explanation: Error.

The attempt to create the indicated MessageQ link driver or server process failed.

User Action: Examine the failure status on the next line to determine if it is a quota or configuration problem.

Failed to translate rights ID "*rights_id*" for DMQ queue g.q

Explanation: Warning.

MessageQ detected an error while performing a SYS\$ASCTOID to validate a requester's privileges.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Failed to wakeup process to complete the queue attach --- PID: "pid"

Explanation: Error.

The COM Server was unable to queue an AST to a process in order to attach the process to a queue.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Fatal error while creating GNT

Explanation: Fatal.

MessageQ detected an error during the creation of the Group Name Table global section.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Fatal error while creating Large LLS Pool

Explanation: Fatal.

MessageQ detected an error during the creation of the large Link List Section or buffer pool global section.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Fatal error while creating MCS

Explanation: Fatal.

MessageQ detected an error during the creation of the message control global section.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Fatal error while creating MRQ Section

Explanation: Fatal.

MessageQ detected an error during the creation of the Multi-reader Queue Global Section.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Fatal error while creating Medium LLS Pool

Explanation: Fatal.

MessageQ detected an error during the creation of the medium Linked List Section or buffer pool global section.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Fatal error while creating Small LLS Pool

Explanation: Fatal.

MessageQ detected an error during the creation of the small Linked List Section or buffer pool global section.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

GNT already exists --- Reinitializing

Explanation: Error.

While attempting to create the Group Name Table global section, MessageQ found that the table already exists.

User Action: None.

Group *g* (*group_name*) OUTBOUND link to system *system_name* is not responding

Explanation: Error.

A communications handshaking or protocol error has occurred on the connection to the indicated group. The remote group is not responding. The group connection will be dropped.

User Action: If the problem persists, contact MessageQ support with a description of the error.

Ignored connect confirm from system *system_name* group *g* due to security match failure

Explanation: Error.

XGROUP_VERIFY has been enabled and the connection confirmation does not match the information found in the Cross-Group Connection Table.

User Action: Correct as necessary.

Ignoring connect request from system *system_name* - group *g* is in link management transition

Explanation: Warning.

The connection request received from the indicated system cannot be processed because the group was caught during in an internal state transition.

User Action: Reattempting the connection, possibly at a later time, should succeed. Repeated occurrences of this warning for the same group may indicate an internal problem which will only get corrected by restarting that group. Contact MessageQ support under this condition.

Ignoring connection for group *g* (*group_name*) from system *system_name* --- link already established

Explanation: Warning.

An incoming connection for a group was received from a system different than the one that it is currently connected to. This is an indication of two groups having the same group number on the same bus.

User Action: Change the group number of system that is the duplicate.

Ignoring connection from system *system_name* for group *g* *group_name* --- connection already on another transport

Explanation: Warning.

An incoming connection was rejected because a link to that group already existed and is owned by another link driver.

User Action: None.

Ignoring LD_REQUEST_ACCEPTED msg from *g.g* --- cannot locate link

Explanation: Warning.

A valid group ownership "REQUEST ACCEPTED" message from the server at the indicated address was received, but there was no such request outstanding.

User Action: Isolated occurrences of this message can be ignored if there are reasons to suspect that the network became inaccessible, or there were connection collisions, just before the message was received.

Ignoring LD_REQUEST_ACCEPTED msg from *g.g* --- unexpected error

Explanation: Error.

An attempt to process a "REQUEST ACCEPTED" response to an ownership transfer failed due to an unexpected error. The group may no longer accept connects.

User Action: Examine the status posted on the next line to determine the error condition encountered, and the corrective action required.

Ignoring LD_REQUEST_DENIED msg from *g.g* --- cannot locate link

Explanation: Warning.

A valid group ownership "REQUEST DENIED" message from the server at the indicated address was received, but there was no such request outstanding.

User Action: Isolated occurrences of this message can be ignored if there are reasons to suspect that the network became inaccessible, or there were connection collisions just before the message was received. Repeated occurrences of this message may indicated a problem. Under this condition, contact MessageQ support.

Ignoring LD_REQUEST_DENIED msg from *g.q* --- unexpected error

Explanation: Error.

An attempt to process a "REQUEST DENIED" response to an ownership transfer failed due to an unexpected error. The group may no longer accept connects.

User Action: Examine the status posted on the next line to determine the error condition encountered and the corrective action required.

Invalid disable queue notification request from *g.q*

Explanation: Warning.

An invalid Queue Notification deregistration request message was received.

User Action: Recheck the request message for validity using script message tracing.

Invalid DMQ\$BUS_GROUP = "*s*"

Explanation: Fatal.

MessageQ detected an invalid setting of the logical DMQ\$BUS_GROUP.

User Action: Correct as necessary.

Invalid queue list request from *g.q*

Explanation: Warning.

An invalid queue list request message was received.

User Action: Recheck the request message for validity using script message tracing.

Invalid Queue Notification Request from *g.q*

Explanation: Warning.

A queue notification registration request message was received that was invalid.

User Action: Recheck the request message for validity using script message tracing.

Invalid msg targeted to *server_name*

+ discarded msg - Src=*g.q* Target=*g.q* class=*n* type=*n*

Explanation: Warning.

The indicated server received a message of a class and type that it does not recognize.

User Action: Correct as necessary.

Invalid X-Group number *g* specified for self

Explanation: Fatal.

Local group number is out of range.

User Action: Correct as necessary.

Invalid X-Group request to declare a temporary SQ - Source=g.q

Explanation: Warning.

A non local process attempted to attach a temporary secondary queue.

User Action: Correct as necessary.

Invalid undeclare SQ request --- SQ #q Source=g.q

Explanation: Error.

A request was made to detach a secondary queue from a process that was invalid, did not own the queue, or it was not a secondary queue.

User Action: Correct as necessary.

Large buffer size is too small, adjusting large buffer size to n

Explanation: Warning.

The Large buffer size provided in the group initialization file is too small. It must be larger than both the small and medium buffers.

User Action: Increase the size of the large buffers in the group initialization file.

Large LLS Pool already exists --- two DMQ COM Servers running

Explanation: Fatal.

While attempting to create the large linked list section (LLS), MessageQ found that the section already exists. This error means that one or more processes have not unmapped the global sections so that the system can delete them. This error could be caused by a process in the debugger, or the user mode ASTs might be disabled.

User Action: Search for processes that are still attached to MessageQ and cause them to exit. You can search for processes using DMQ\$EXE.DMQ\$SCAN_SYSTEM_FOR_DMQ.COM.

Link Driver has terminated

Explanation: Warning.

The link driver has terminated execution.

User Action: Examine the previous messages, and the log file, for more information concerning the action.

Link Management request from g.q rejected

Explanation: Error.

A Link Management request message, generated by a user program, was received either from an nonprivileged user or from another group.

User Action: Investigate proper authorization of request.

Listener is down - network not currently accessible

Explanation: Error.

The transport listener cannot declare itself due to the network being inaccessible. The transport is no longer available for use in group connections.

User Action: Bringing the network up again will automatically cause the link driver to restart itself and all connections.

Load complete message received from *g.g* --- Ignored

Explanation: Warning.

A non local load complete message was received.

User Action: Correct as necessary.

Local host address lookup failed

Explanation: Fatal.

The link driver cannot execute because it cannot find its own system address in the applicable network database.

User Action: Examine the network database, and the MessageQ initialization file.

Local host name lookup failed

Explanation: Fatal.

The link driver failed to locate the local host name in the network database.

User Action: Examine the network database, and the MessageQ initialization file.

Local system is *system_name*, transport addr *s*

Explanation: Informational.

Identifies the system and transport address of the link driver starting up.

User Action: None.

Local system *system_name* is not listed for this group *g* under this transport

Explanation: Fatal.

The specified system name is not listed for this group under this transport in the groups initialization file.

User Action: Update the group initialization file and restart the group.

Local system name is too long

Explanation: Fatal.

Maximum DECnet system name is 6 characters.

User Action: Modify the group initialization file and restart the group.

Logical DMQ\$SET_GBLSEC_PROT value "s" is invalid

Explanation: Fatal.

The format of the DMQ\$SET_GBLSEC_PROT logical name is invalid.

User Action: Correct as necessary.

**** LOW BUFFER WARNING --- LARGE FREE LIST --- TOTAL:n CURR:n
WARN:n ****

Explanation: Warning.

The number of large free buffers available has reached the warning level.

User Action: You might need to adjust send rate, total number of large buffers available, or receive byte quota on the message queues.

**** LOW BUFFER WARNING --- MEDIUM FREE LIST --- TOTAL:n CURR:n
WARN:n ****

Explanation: Warning.

The number of medium free buffers available has reached the warning level.

User Action: You might need to adjust send rate, total number of medium buffers available, or receive byte quota on the message queues.

**** LOW BUFFER WARNING --- SMALL FREE LIST --- TOTAL:n CURR:n
WARN:n ****

Explanation: Warning.

The number of small free buffers available has reached the warning level.

User Action: You might need to adjust send rate, total number of small buffers available, or receive byte quota on the message queues.

MCS already exists --- two DMQ COM Servers running

Explanation: Fatal.

While attempting to create the message control section (MCS), MessageQ found it to already exist. This error means that one or more processes have not unmappped the global sections so that the systems can delete them. This error could be caused by a process in the debugger or because the user mode ASTs are disabled.

User Action: Search for processes that are still attached to MessageQ and cause them to exit. You can search for processes using DMQ\$EXE.DMQ\$SCAN_SYSTEM_FOR_DMQ.COM.

Medium buffer size is too small, adjusting medium buffer size to n

Explanation: Warning.

The medium buffer size provided in the group initialization file is too small. It must be larger than the small buffer size.

User Action: Increase the size of the medium buffers in the group initialization file.

Medium LLS Pool already exists --- two DMQ COM Servers running

Explanation: Fatal.

While attempting to create the medium linked list section (LLS), MessageQ found that it already exists. This error means that one or more processes have not unmapped the global sections so that the system can delete them. This error could be caused by a process in the debugger, or because the user mode ASTs are disabled.

User Action: Search for processes that are still attached to MessageQ and cause them to exit. You can search for processes using `DMQ$EXE.DMQ$SCAN_SYSTEM_FOR_DMQ.COM`.

Message lost (undeliverable)

+ discarded msg --- Src=*g.q* Target=*g.q* class=*n* type=*n*

Explanation: Error.

MessageQ could not deliver the message to its destination queue. The UMA will be taken.

User Action: Correct as necessary.

Message received from bus returned error

Explanation: Error.

An attempt to receive a message from the MessageQ message bus returned an unexpected error.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Msg from system *system_name* is larger than negotiated maximum for group *g* (*group_name*)

Explanation: Error.

A message was received from the indicated system that exceeds the maximum size supported for the group. The connection to the group will be dropped. This message indicates a mismatched configuration between the two groups, or an attempt to use link management services on a group that has a message length maximum below the minimum requirements.

User Action: Change the group configuration to support a size maximum large enough to handle the message.

MONITOR terminate requests are disabled

Explanation: Warning.

A user attempted to shut down the COM Server by using the "Kill COM

Server" MONITOR command. This command has been explicitly disabled by setting the ACCEPT_KILL_CMD to NO in the Profile section of the group initialization file.

User Action: Correct as necessary.

Msg lost (bad target address)

+ discarded msg --- Src=*g.g* Target=*g.g* class=*n* type=*n*

Explanation: Error.

MessageQ could not deliver the message to its destination queue because the destination queue could not be found. The UMA will be taken.

User Action: Correct as necessary.

Network listener has been restarted

Explanation: Informational.

The link driver has automatically recovered from previously being unable to declare its listener (most probably due to the network being inaccessible).

User Action: None.

Network listener is down!

Explanation: Error.

The transport listener cannot declare itself due to an unknown error condition. The transport is no longer available for use in group connections.

User Action: Examine the error status posted on the next line to determine the network problem. Correcting the problem and bringing the network up again will automatically cause the link driver to restart itself and all connections.

Network is down - all links will be dropped

Explanation: Error.

The network has become inaccessible and all existing connections to groups are being dropped. The link driver will retry connections if the network becomes accessible again.

User Action: Restart the network.

Network protocol error occurred for message queue group *g* (*group_name*)

Explanation: Error.

A DECnet protocol error occurred on the specified link to the remote group.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

No available temporary queues - Unable to allocate temp SQ - Source=*g.g*

Explanation: Warning.

While attempting to allocate a temporary secondary queue to a process,

MessageQ found the pool of available queues to be empty.

User Action: Increase the number of temporary queues by lowering the `FIRST_TEMP_QUEUE` value in the groups initialization file.

No available temporary queues --- Unable to allocate queue --- Source=*g.g*

Explanation: Warning.

While attempting to allocate a temporary queue to a process, MessageQ found the pool of available queues to be empty.

User Action: Increase the number of temporary queues by lowering the `FIRST_TEMP_QUEUE` value in the groups initialization file.

No memory for buffer for message queue group *g* (*group_name*)

Explanation: Error.

An error was detected by DECnet software while attempting to create buffers for use by the specified link.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem or a system resource problem.

Protocol error on OUTBOUND link for group *g* (*group_name*) to system *system_name*

Explanation: Error.

A communications handshaking or protocol error has occurred on the connection to the indicated group. The group connection will be dropped.

User Action: If the problem persists, contact MessageQ support with a description of the error.

Rejected connect attempt to/for group *g* - Ownership transfer denied by *g.g*

Explanation: Warning.

The connection attempt was rejected because a request for group ownership transfer was denied by the server at the indicated address.

User Action: None.

Rejected connect request for group *g* - system could not be determined

Explanation: Error.

The connection request cannot be accepted because the remote system could not be determined. This may indicate an error in the MessageQ initialization file, or that the system is not listed in the appropriate network database.

User Action: Examine and correct the MessageQ initialization file(s) or network database.

Rejected connect request for group *g* from system *system_name* (*system_address*) - address disagrees with network database

Explanation: Error.

A connection attempt was received from a system whose name and address do not match the information in the applicable network database. The connection request will be rejected.

User Action: Check that the system names and addresses in the network database agree with each other, and with the MessageQ initialization file(s).

Rejected connect request for group *g* from system *system_name* - outbound attempt failed

Explanation: Error.

An outbound connection attempt to the indicated system and group failed due to an unknown error condition.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Rejected connect request for group *g* from system *system_name* (*system_address*) - system not found in network database

Explanation: Warning.

A connection attempt was received from a system that does not exist in the applicable network database.

User Action: The MessageQ initialization file(s) may be in error, or the system may need to be added in the network database.

Rejected connect request from system *system_name* due to security match failure (group unknown)

Explanation: Error.

XGROUP_VERIFY has been enabled and the connect request does not match the information found in the Cross-Group Connection Table.

User Action: Correct as necessary.

Rejected connect request from system *system_name* due to wrong bus

Explanation: Error.

The connection attempt was rejected because it came from a different MessageQ bus.

User Action: Correct the environment so that the group connections are for the same bus.

Rejected connect request from system *system_name* group *g* due to security match failure

Explanation: Error.

XGROUP_VERIFY has been enabled and the connect request does not match the information found in the Cross-Group Connection Table.

User Action: Correct as necessary.

Rejected connect request from system *system_name* for group *g* (*group_name*) - Connection already in progress by *b.g*

Explanation: Warning.

The connection request received from the indicated system was rejected because the connection is already in progress.

User Action: None.

Rejected connect request from system *system_name* for group *g* (*group_name*) - connection currently in progress

Explanation: Warning.

The connection request received from the indicated system was rejected because the connection is currently in progress.

User Action: None.

Rejected connect request from system *system_name* for group *g* (*group_name*) - connection still in progress

Explanation: Warning.

The connection request received from the indicated system was rejected because the connection is still in progress.

User Action: None.

Rejected connect request from system *system_name* group *g* - Group disabled

Explanation: Warning.

An incoming connect request was denied because connections are currently disabled. This can be due to the group being disabled on startup or by way of a Link Management message disconnecting the link.

User Action: When ready to allow links to this group, send a Link Management message to enable connections.

Rejected connect request from system *system_name* for group *g* (*group_name*) - group is already connected

Explanation: Warning.

A connection request was received from the indicated system for a group that is already connected (possibly to a different transport).

User Action: None.

Rejected connect request from system *system_name* for group *g* (*group_name*) - group is disabled

Explanation: Warning.

An incoming connect request was denied because connections are currently disabled. This can be due to the group being disabled on startup or by way of a Link Management message disconnecting the link.

User Action: When ready to allow links to this group, send a Link Management message to enable connections.

Rejected connect request from system *system_name* (*system_address*) - Improper Protocol response received

Explanation: Error.

A connection attempt from the indicated system was rejected because of a communications protocol failure. This may indicate a non-linkdriver program (perhaps a non-MessageQ program), or an incompatible MessageQ system is attempting the connection.

User Action: Investigate the system and program attempting the connection.

Rejected connect request from system *system_name* for group *g* *group_name* -Link Driver or network is shutting down

Explanation: Error.

The connect request is ignored because either the link driver, the network, or the system is shutting down.

User Action: Correct as necessary.

Rejected connect request from system *system_name* for group *g* (*group_name*) - Link is currently in transition

Explanation: Warning.

The connection request from the indicated system was rejected because an interruptable link transition was already in progress for that group.

User Action: Reattempting the connection, possibly at a later time, should succeed. Repeated occurrences of this warning for the same group may indicate an internal problem which will only get corrected by restarting that group. Contact MessageQ support under this condition.

Rejected connect request from system *system_name* for group *g* (*group_name*) - Lost conflict resolution

Explanation: Warning.

The connection request received from the indicated system was rejected due to a different transport currently owning the group.

User Action: None.

Rejected connect request from system *system_name* for group *g* (*group_name*) - read link is currently in transition

Explanation: Warning.

The connection request received from the indicated system was rejected because the read link is currently in transition.

User Action: None.

Rejected connect request from system *system_name* for group *g* (*group_name*) - write link is currently in transition

Explanation: Warning.

The connection request received from the indicated system was rejected because we own this group, and there's already an outbound, and that outbound is currently shutting down.

User Action: None.

Rejected connect request from system *system_name* (*system_address*) - protocol not supported

Explanation: Error.

A connection attempt was received from a system running a MessageQ release that is incompatible with the link driver.

User Action: None.

Remote system not listed for this group/transport

Explanation: Error.

The connection request is rejected because the remote group is not listed for this group or transport and XGROUP_VERIFY is YES.

User Action: Correct as necessary.

***Server_name* has exited**

Explanation: Warning.

The indicated link driver or server process has terminated.

User Action: Determine the reason for the termination by examining the link driver or server log files in DMQ\$LOG:.

Skipping connect to group *g* (*group_name*) - Failed to send external ownership request to *b.g*

Explanation: Error.

A connection to the indicated group could not be attempted because of a failure to send an external ownership request to the specified group.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Skipping connect to group \mathcal{G} (*group_name*) - No outbounds possible or allowed

Explanation: Informational.

A connect attempt to the indicated group was not possible or not supported.

User Action: None.

Skipping connect to group \mathcal{G} (*group_name*) - System address cannot be determined

Explanation: Informational.

A connection cannot be attempted to the indicated group because the system address could not be determined. This may indicate an error in the MessageQ initialization file, or that the named system is not listed in the appropriate network database.

User Action: Examine and correct the MessageQ initialization file(s) or network database.

Skipping connect to group \mathcal{G} (*group_name*) - unexpected error encountered

Explanation: Error.

A connection to the indicated group could not be attempted due to an unexpected error condition.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

Small LLS Pool already exists --- two DMQ COM Servers running

Explanation: Fatal.

While attempting to create the small Linked List Section, MessageQ found them to already exist. This means that one or more processes have not unmapped the global sections so that the system can delete them. This may be due to a process in the debugger or because the user mode ASTs are disabled.

User Action: Search for processes that are still attached to MessageQ and cause them to exit. You can search for processes using

`DMQ$EXE.DMQ$SCAN_SYSTEM_FOR_DMQ.COM.`

Starting *server_name*

Explanation: Informational.

The specified server has been spawned by the COM server.

User Action: None.

***Server_name* Initialized**

Explanation: Informational.

The link driver or server specified has successfully started running.

User Action: None.

SYSS\$CRELNM failed to create *logical_name* in lnm tbl table_name

Explanation: Warning.

The call to SYSS\$CRELNM returned a failure while attempting to create the logical name in the specified table.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

SYSS\$CRELNM failed to create DMQ\$VERSION

Explanation: Warning.

The call to SYSS\$CRELNM returned a failure while attempting to create the DMQ\$VERSION logical name.

User Action: Examine the failure status posted on the next line to determine if it is due to a process quota problem.

This linkdriver is already running --- exiting

Explanation: Error.

This link driver is already running, or previously terminated abnormally, and cannot be restarted.

User Action: If it is not an accidental attempt to run two copies of the same link driver, you must restart the group to clear the condition.

This transport is not listed for this group *g*

Explanation: Fatal.

The link driver cannot execute because the applicable transport does not appear in any of the systems listed for the local group.

User Action: The MessageQ group initialization file will need to be corrected; otherwise, the transport will not be available for use.

Timed out waiting for link driver(s) to start

Explanation: Error.

One or more Link Drivers failed to initialization within the timeout period.

User Action: Determine the cause by checking the link drivers log file to see if there was a startup failure or messaging error. It also could be due to system load which can be determined by comparing time-stamp on the link driver "Startup completed" message and this error in the COM Server log file.

The timeout period for server startup can be modified by providing a server startup time parameter to the DMQ\$STARTUP.COM command line.

Transport address doesn't match xgroup entry

Explanation: Error.

The transport address does not match the address in the xgroup entry table and

XGROUP_VERIFY is YES.

User Action: Correct as necessary.

Transport address is already in use by another process

Explanation: Fatal.

The transport listener cannot declare itself because the transport address (port number), as defined in the MessageQ initialization file for the local group, is already being used by another process.

User Action: Free up the address, or change the MessageQ initialization file(s) to use a different transport address.

Transport not supported by group

Explanation: Error.

The connection request is rejected because the specified transport is not supported by this group and XGROUP_VERIFY is YES.

User Action: Correct as necessary.

TP PROTOCOL ERROR: invalid substate n on INBOUND link to group *g* (*group_name*)

Explanation: Error.

A communications handshaking or protocol error has occurred on the connection to the indicated group. The group connection will be dropped.

User Action: If the problem persists, contact MessageQ support with a description of the error.

TP PROTOCOL ERROR: invalid substate n on OUTBOUND link to group *g* (*group_name*)

Explanation: Error.

A communications handshaking or protocol error has occurred on the connection to the indicated group. The group connection will be dropped.

User Action: If the problem persists, contact MessageQ support with a description of the error.

Unable to deassign DECnet data I/O channel for group *g* in XDECNET

Explanation: Warning.

MessageQ detected an error when attempting to deassign an I/O channel for a remote group.

User Action: Verify whether an earlier link error is causing this problem.

Unable to deassign DECnet OOB channel for group *g* in XDECNET

Explanation: Warning.

MessageQ detected an error when attempting to deassign a DECnet OOB

channel for a remote group.

User Action: Verify whether an earlier link error is causing this problem.

Unable to declare temporary SQ --- Source=*g.q*

Explanation: Warning.

An error was detected while attempting to attach a temporary secondary queue.

User Action: Examine the failure status posted on the next line to determine the cause of the problem.

Unable to locate DECnet TCB for group *g*

Explanation: Error.

MessageQ was unable to locate a DECnet entry for the group in the transport control block.

User Action: Make sure DECnet is specified for the group in the group initialization file.

Unable to locate local group definition --- Cross-group disabled

Explanation: Error.

MessageQ was unable to locate the entry in the cross-group table that defines the local group. This is required to be present when cross-group support has been enabled.

User Action: Edit the group initialization file (DMQ\$INIT.TXT) and either define the local group in the cross-group table or disable cross-group support.

Unable to return link management response to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to send a link management response message.

User Action: Examine the failure status posted on the next line to determine the cause of the problem.

Unable to return list of groups to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to send a group list message.

User Action: Examine the failure status posted on the next line to determine the cause of the problem.

Unable to return list of processes to *g.q*

Explanation: Warning.

MessageQ detected an error while attempting to send a process list message.

User Action: Examine the failure status posted on the next line to determine the cause of the problem.

Unable to return list of queues to g.q

Explanation: Warning.

MessageQ detected an error while attempting to send a queue list message.

User Action: Examine the failure status posted on the next line to determine the cause of the problem.

Unable to return queue notify disable response to g.q

Explanation: Warning.

MessageQ detected an error while attempting to send a queue notify disable response message.

User Action: Examine the failure status posted on the next line to determine the cause of the problem.

Unable to send link arbitration msg for group g to g.q

Explanation: Error.

An attempt to respond to a request for transfer of group ownership failed.

User Action: Examine the status posted on the next line to determine the error condition encountered and the corrective action required.

Unable to send message to target - exceeded x-group buffer maximum

+ discarded msg - Src=g.q Target=g.q class=n type=n

Explanation: Error.

An attempt was made to send a message that is larger than the receiving groups GROUP_MAX_MESSAGE_SIZE.

User Action: Increase the receiving groups GROUP_MAX_MESSAGE_SIZE in the groups initialization file.

Unblock msg lost (buffer allocation failed)

Explanation: Warning.

An attempt to send an unblock message failed because a buffer could not be allocated. No more memory.

User Action: Increase page file quotas for the server and restart group.

XGroup not enabled --- program exiting

Explanation: Fatal.

The ENABLE_XGROUP parameter in the group initialization file is set to NO.

User Action: Set the ENABLE_XGROUP parameter in the group initialization file to YES. Restart the group.

X-Group write failed to group *g* (*group_name*)

+ Discarded msg --- Src=*g.g* Target=*g.g* class=*n* type=*n*

Explanation: Warning.

A cross-group connection request between the listed groups failed.

User Action: None.

X-Group write failed to group *g* (*group_name*)

+ Message lost - protocol translation or UMA failure

Explanation: Error.

An attempt to send a x-group message to a remote system failed due to an unknown error condition, and the message cannot be recovered.

User Action: Examine the failure status posted on the next line to determine the error condition encountered, and the corrective action required.

X-Group write pool overflow for group *g*

Explanation: Warning.

One or more programs have generated message traffic to a remote message queuing group at a rate faster than can be sent. This traffic eventually results in cross-group buffer pool overflowing and therefore requires the Undeliverable Message Action (UMA) to be taken.

User Action: Change the delivery mode to one that will inherently cause pacing, increase the cross-group buffer pool so that it can absorb the temporary burst of messages, or embed a pacing protocol in the application dialogue.

Additional Information on Link Drivers Error Status Reporting

The DECnet and TCP/IP Link Drivers share a common error status reporting translation routine. For example, `SS$_REJECT` would be logged, however the actual error is transport dependent: `MSG$_REJECT` for DECnet, and `ECONNREFUSED` for TCP link drivers. Refer to the table below and further to the documentation reference for additional information.

Displayed Status	TCP/IP	DECnet
SS\$_REJECT	ECONNREFUSED - Connection refused No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host.	MSG\$_REJECT - Connect Reject The object at remote host was not available or has actively rejected the connection.
SS\$_UNREACHABLE	EHOSTUNREACH - No route to host ENETUNREACH - Network is unreachable A socket operation was attempted to an unreachable host or network.	MSG\$_NODEINACC - Node has become inaccessible
SS\$_TIMEOUT	ETIMEDOUT - Connection timed out. A "connect()" request failed because the connected party did not properly respond after a period of time. (The timeout period is dependent on the communication protocol.)	MSG\$_TIMEOUT - Connect timeout
SS\$_ABORT	(Not Used)	MSG\$_ABORT - Partner aborted link
SS\$_SHUT	ESHUTDOWN - Can't send after socket shutdown ENETDOWN - Network is down A socket operation encountered a dead network.	MSG\$_NETSHUT - Network shutting down

Reference *DEC TCP/IP Services for OpenVMS System Service and C Socket Programming* Table 6-1: errno Values for further error message details for TCP/IP. Reference *DECnet for OpenVMS Networking Manual* Chapter 8 for further error message details for DECnet.

MRS Server Error Log Messages

This section lists and describes MRS Server error log messages. Use the following Key to understand conventions used in this section:

Key:

- *g* = Group ID decimal value
- *q* = Queue number decimal value
- *n* = A numeric value
- *x* = Hexadecimal value
- *s* = Character string value
- *Italic* text represents information that can vary in log messages.

JRN_E_DLJ_FILES, NUM_DLJ_AREAS quota exceeded

Explanation: Error.

An error occurred while the Journal server was attempting to create a new DLJ file. The maximum number of DLJ files has been reached.

User Action: Increase NUM_DLJ_AREAS in group initialization file or increase AREA_SIZE to decrease the number of files required.

JRN_E_PCJ_FILES, NUM_PCJ_AREAS quota exceeded

Explanation: Error

An error occurred while the Journal server was attempting to create a new PCJ file. The maximum number of PCJ files has been reached.

User Action: Increase NUM_PCJ_AREAS in group initialization file or increase AREA_SIZE to decrease the number of files required.

JRN_E_INIT, JRN Server did not initialize properly

Explanation: Error.

The Journal Server did not complete its initialization process.

User Action: Check EVL and JRN server logs for errors.

JRN_E_INVDIRP, Invalid recovery protocol:n for msg (x,x) with DIP:delivery_mode

Explanation: Error.

The MRS Server will reject messages that are targeted to a 1.0 recovery protocol group if the specified delivery mode is not supported by the receiving group. Example, DM_CONF delivery mode is only supported by version 3.x and up, but is not supported by version 2.1 groups.

User Action: Either upgrade the receiving group to a current version of the MessageQ software, or modified the delivery mode in the application code.

JRN_E_INVRP, Invalid recovery protocol:n for msg (x,x)

Explanation: Error.

MessageQ detected an invalid recovery protocol value in the message header.

User Action: Contact BEA MessageQ support.

JRN_E_JRNCLS, Error closing Journal file filename

Explanation: Error

An error occurred while the Journal Server was closing a journal file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

JRN_E_JRNERR, Unexpected error in Journal file filename, status=n

Explanation: Error.

An unexpected error was encountered by the Journal Server in accessing the journal file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

JRN_E_JRNMSGTOOBIG, Journal record is too large for Journal file

Explanation: Error.

Message is too large to fit into the journal file.

User Action: Increase the AREA_SIZE in the MRS section of the group initialization file.

JRN_E_JRNOPN, Error opening Journal file filename

Explanation: Error

An error occurred while the Journal Server was attempting to open a journal file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

JRN_E_JRNTRUNC, Incomplete Journal record detected filename

Explanation: Error

The Journal Server detected an incomplete journal record in the specified file.

User Action: Check the file system and PCJ/DLJ files for disk errors. Use the Manger Utility to dump the contents of the journal file for analysis.

JRN_E_JRNWR, Error writing Journal file filename

Explanation: Error.

MessageQ detected an error when writing to the journal file.

User Action: Check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_AM_D, Attempt to add duplicate msg (x,x) to ML

Explanation: Error.

The MRS Server attempted to add a message to the MRS message list (ML) that was already in the ML. MRS will remove the original message from the ML and the DQF and replace it with the new message.

User Action: None.

MRS_E_AM_S, Message data base full. Increase page file quota.

Explanation: Error.

The MRS Server attempted to add a message to the MRS message list that was full. MRS failed to expand the message list.

User Action: Increase the page file quota for the MRS server in the `DMQ$SET_SERVER_LOGICALS.COM` file.

MRS_E_AP_S, Queue list full. Increase NUM_QUEUES and page file quota.

Explanation: Error.

The MRS Server attempted to add a queue to the MRS queue list that was full. MRS failed to expand the queue list.

User Action: Increase page file quota for MRS server in the `DMQ$SET_SERVER_LOGICALS.COM` file and increase the `NUM_QUEUES` parameter in the group initialization file (`DMQ$INIT.TXT`).

MRS_E_BADJRN, g,q sent invalid confirm mode: n

Explanation: Error.

Invalid parameter passed into the `pams_confirm_msg` force journal field.

Valid values are `PDEL_DEFAULT_JRN`, `PDEL_FORCE_JRN` and `PDEL_NO_JRN`

User Action: Recheck application code and documentation.

MRS_E_BADLOGIC, UNEXPECTED ERROR error_text

Explanation: Error.

An unexpected code path or condition occurred. The text of the message will provide further information.

User Action: Contact BEA support for more information.

MRS_E_CIOERR, C I/O error_text : file: filename

Explanation: Error.

An error was returned by C language I/O support. The text of the message will provide further information.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_CNF_P, Attempt by *b.g* to confirm unk. msg (*x,x*)

Explanation: Error.

The message sequence number parameter supplied to the `pams_confirm_msg` function call is not in the MRS Server message list. Message has either already been confirmed or the message sequence number is invalid.

User Action: Correct the programming error in the application program that calls `pams_confirm_msg`.

MRS_E_CHKPTREQFAIL, Checkpoint file operation request failed

Explanation: Error.

A failure occurred while attempting to open the checkpoint file.

User Action: Check other error messages logged. The checkpoint file is probably locked by another user.

MRS_E_CHN, %RMS-F-CHN, Assign channel system service request failed

Explanation: Error.

MessageQ detected an error when attempting to create a journal file. No more I/O channels available.

User Action: Check the system channel count and increase the size of the journal files to decrease the number of dynamic file creations.

MRS_E_CLEANONCLOSE, Journal cleanup on close enabled by logical DMQ\$MRS_CLEANUP_ON_CLOSE

Explanation: Informational.

Journal file cleanup (DQF and SAF) on close has been enabled. This allows empty DQF or SAF files to be deleted on closure if they are empty.

User Action: None.

MRS_E_DELDUP, Failed to remove stale msg (*x,x*) from filename

Explanation: Error.

MessageQ detected an error while attempting to delete a duplicate message from the DQF.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DEQ, Error removing msg (*x,x*) from DQF

Explanation: Error.

MessageQ detected an error while deleting a message from the DQF.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQF_ALLOC, Error allocating space for DQF list

Explanation: Fatal.

An attempt to allocate virtual memory by the MRS Server failed. This failure prevented the MRS Server from starting.

User Action: Increase the MRS servers page file quota in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_DQFBYPASS, I/O Error caused beeps of DQF file: *filename*

Explanation: Error.

An I/O error while attempting to access the DQF file has resulted in the file being bypassed. Normally a bypassed DQF file will be renamed with the extension `.DERR`.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFCLS, Error closing DQF file *filename*

Explanation: Error

An error occurred while MRS was closing a DQF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFDEL, Error deleting DQF file *filename*, status=*x*

Explanation: Error.

An error occurred while MRS was attempting to delete an empty DQF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFERR, Unexpected error in DQF file *filename*

Explanation: Error.

An unexpected error was encountered by MRS in accessing the DQF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQF_FILES, DQF_FILES quota exceeded

Explanation: Error

An error occurred while MRS was attempting to create a new DQF file. The maximum number of DQF files has been reached.

User Action: Increase `NUM_DQF_AREAS` in group initialization file or increase `AREA_SIZE` to decrease the number of files required.

MRS_E_DQFIDXERR, Unexpected error building DQF list

Explanation: Error

An error occurred while attempting to access the Process List.

User Action: Contact BEA MessageQ support.

MRS_E_DQFIODEL, IO error prevents use of DQF file filename

Explanation: Error

An error occurred while MRS was attempting a file operation on the DQF file. The file has been renamed with extension `.DERR`.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFMSGDEL, Error Deleting message from DQF file filename

Explanation: Error.

MessageQ detected an error while deleting a message from the DQF.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFMSGTOOBIG, DQF message is too large for DQF file filename

Explanation: Error.

Message is too large to fit into the specified DQF file.

User Action: Increase the `AREA_SIZE` in the MRS section of the group initialization file.

MRS_E_DQFOPN, Error opening DQF file filename, status= n

Explanation: Error

An error occurred while MRS was attempting to open a DQF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFRD, Error reading DQF file filename

Explanation: Error

An error occurred while MRS was attempting to read a DQF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_DQFTRUNC, Incomplete DQF record detected filename

Explanation: Error

MRS detected an incomplete DQF record. Either a write error resulted in the messages failure to make it to disk, or a delete error resulted in the failure to completely remove the message from disk.

User Action: Check the file system and DQF files for disk errors. Use the Manger Utility to dump the contents of the DQF file for analysis.

MRS_E_DQFW, DQF write error

Explanation: Error.

MessageQ detected an error while writing a message to the DQF.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_EXTENDERR, Journal file filename could not be extended, default AREA_SIZE is too small.

Explanation: Error.

MessageQ failed to extend the journal file.

User Action: Increase AREA_SIZE in group initialization file.

MRS_E_FREE_ALLOC, Error allocating space for FREE file list

Explanation: Fatal.

An attempt to allocate virtual memory by the MRS Server failed. This failure prevented the MRS Server from starting.

User Action: Increase the MRS servers page file quota in DMQ\$SET_SERVER_QUOTAS.COM.

MRS_E_FREECLS, Error closing FREE file

Explanation: Error.

An error occurred while MRS was closing a FREE file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_FREECREATE, Error creating FREE file filename, status=n

Explanation: Error.

An error occurred while attempting to create a FREE file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_FREEDEL, Error deleting FREE file filename, status=n

Explanation: Error.

MessageQ detected an error while attempting to delete a FREE file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_FREE_FILES, FREE_FILES quota exceeded

Explanation: Error.

An error occurred while MRS was attempting to create a new FREE file. The maximum number of FREE files has been reached.

User Action: Increase NUM_DQF_AREAS in group initialization file or increase

AREA_SIZE to decrease the number of files required. The number of FREE files is based on the number of DQF files.

MRS_E_INIT, MRS did not initialize properly

Explanation: Error.

The MRS Server did not complete its initialization process.

User Action: Check EVL and MRS server logs for errors.

MRS_E_INVGRP, Invalid recovery protocol:n for group: g

Explanation: Error.

The MRS Server encountered an invalid recovery protocol for the specified group.

User Action: Contact BEA MessageQ support.

MRS_E_INVPS, Invalid process state in module *module_name* process *state*

Explanation: Error.

The MRS Server encountered an invalid process state in the specified module.

User Action: Contact BEA MessageQ support.

MRS_E_INVRP, Invalid recovery protocol:n for msg (x,x)

Explanation: Error.

The MRS Server encountered an invalid recovery protocol for the specified message.

User Action: Contact BEA MessageQ support.

MRS_E_JRNFAIL, Error fetching (x,x) from DQF prior to PCJ

Explanation: Error.

An I/O error occurred while MessageQ was retrieving a message from the DQF in response to a `pams_confirm_msg` call.

User Action: Check disk quotas. Check the file system and DQF files for disk errors.

MRS_E_LATECONF, Late confirm from b.g for (x,x); value x

Explanation: Error.

This messages is only seen on MRS servers that are talking 1.0 recovery protocol. It is an indication that the MRS server received and `ENQ_STATUS_MSG` for a message that was no longer on the ML. This is an indication that the user timed out before the message was successfully enqueued to the target MRS server.

User Action: Increase the timeout parameter to `pams_put_msg`.

MRS_E_LRBUFTOOSMALL, Message in filename is larger (n bytes) than the LARGE buffer size configured for this group (n bytes)

Explanation: Error.

The MRS Server is attempting to process a message that is too large for the internal buffer sizes. This may occur if some large messages were stored in a journal file and then the maximum group message sizes were lowered and the group restarted. MRS cannot process the messages that are stored in the journal file because they are too large for the current buffer.

User Action: Increase the `GROUP_MAX_MESSAGE_SIZE` in the group initialization file and restart the group. If that does not fix the problem then contact BEA MessageQ support.

MRS_E_MALLOC, Memory allocation failed

Explanation: Error.

The MRS Server was unable to allocate virtual memory.

User Action: Increase the MRS servers page file quotas in `DMQ$SET_SERVER_QUOTAS.COM`

MRS_F_MALLOC, Memory allocation failed

Explanation: Fatal.

The MRS Server was unable to allocate virtual memory.

User Action: Increase the MRS servers page file quota in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_ML_ADD, Could not add message to ML list

Explanation: Error.

MessageQ was unable to add messages to the message list.

User Action: This message will be accompanied by other error messages that will indicate the specific resource allocation failure.

MRS_E_ML_ALLOC, ML memory alloc failed

Explanation: Fatal.

An attempt to allocate virtual memory by the MRS Server failed. This failure prevented the MRS server from starting.

User Action: Increase the MRS servers page file quota in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_MLAGE, Msg (x,x) to b.g stuck on ML

Explanation: Error.

The MRS Server encountered an internal MRS error.

User Action: Contact BEA MessageQ support.

MRS_E_MLNF, Message (x,x) not ML

Explanation: Error.

The MRS Server code not locate the message on the message list.

User Action: Contact BEA MessageQ support if the problem does not correct itself.

MRS_E_MSGERR, Cant send msg (x,x) b.g to b.g, len:n

Explanation: Error.

The MRS Server encountered an error while attempting to send a message.

The text of message will indicate the error. If the error is PAMS_REMQUEFAIL, the queuing resources of the group were exceeded or insufficient global memory buffers were found to contain a particular message request.

User Action: Increase the buffer pool space for the group or modify the sender program to manage message flow control.

MRS_E_MSGTOOBIG, Message exceeds maximum allowed size

Explanation: Error.

The MRS Server is attempting to process a message that is too large for the internal buffer sizes. This may occur if some large messages were stored in a journal file and then the maximum group message sizes were lowered and the group restarted. MRS cannot process the messages that are stored in the journal file because they are too large for the current buffer.

User Action: Increase the GROUP_MAX_MESSAGE_SIZE in the group initialization file and restart the group. If that does not fix the problem then contact BEA MessageQ support.

MRS_E_NOMEM, Memory allocation failure during file I/O

Explanation: Error.

An attempt to allocate virtual memory by the MRS Server failed.

User Action: Increase the MRS servers page file quota in DMQ\$SET_SERVER_QUOTAS.COM.

MRS_E_NORETAIN, Message not retained in holdaside buffer

Explanation: Error.

An error prevented the allocation of an internal buffer to hold the message aside for a timed retry.

User Action: Increase the MRS servers page file quota in DMQ\$SET_SERVER_QUOTAS.COM.

MRS_E_NTA, MRS notify alloc failed

Explanation: Error.

An attempt to reallocate a message buffer failed.

User Action: Contact BEA MessageQ support.

MRS_E_PAMS_SEND, Error sending msg to (b,g) class:n type:n

Explanation: Error.

The MRS Server encountered an error while attempting to send a message.

The text of message will indicate the specific error. If the error is

PAMS_REMQUEFAIL, the queuing capacity of the group was exceeded or insufficient global memory buffers were found to contain a particular message request..

User Action: Increase the buffer pool space or modify the sender program to manage message flow control.

MRS_E_PCJFAIL, Error writing (x,x) to PCJ

Explanation: Error.

Message failed to be written to the PCJ file.

User Action: Verify that the Journal Server is running. Check EVL and JRN server logs for errors. Verify that ENABLE_JRN parameter is set to YES in the group initialization file.

MRS_E_PL_ALLOC, PL memory alloc failed

Explanation: Fatal.

An attempt to allocate virtual memory by the MRS Server failed. This failure prevented the MRS Server from starting.

User Action: Increase the MRS servers page file quota in DMQ\$SET_SERVER_QUOTAS.COM.

MRS_E_PLEV, PL entry vanished

Explanation: Error.

Process entry is not in the process list. This is an unexpected error.

User Action: Contact BEA MessageQ support.

MRS_E_PL_MISMATCH, Unexpected PL_idx for b,g found: n, expected:n

Explanation: Error.

The MRS Server detected an internal PL index mismatch.

User Action: Contact BEA MessageQ support.

MRS_E_RB_ALLOC, Ring Buffer memory alloc failed

Explanation: Error.

An attempt to allocate virtual memory by the MRS Server failed.

User Action: Increase the MRS servers page file quota in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_RCV, PAMS receive error

Explanation: Error.

The MRS Server encountered an error while attempting to read a message from the MRS server queue. If the error is `PAMS__EXHAUSTBLKS`, the queuing capacity of the group has been exceeded.

User Action: Increase the MRS servers pool quotas and check the MRS servers page file quotas in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_RDQFD, Error posting READ_DQF_D for *b.g*

Explanation: Error.

The MRS Server failed to post the `READ_DQF_D` message for the specified group.

User Action: None. The MRS Server will retry again later.

MRS_E_RENAME, Failed to rename file from *filename* to *filename*

Explanation: Error.

The MRS failed to rename a journal file. This is a C I/O error.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_RMA, MRS reply alloc failed

Explanation: Error.

An attempt to reallocate a message buffer failed.

User Action: Contact BEA MessageQ support.

MRS_E_RMS, File system error: *error_text*

Explanation: Error.

An RMS I/O call returned an error. The text of the message will provide further information.

User Action: Depending upon text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_RSLOW, *s* restart activity timeout for *b.g*

Explanation: Error.

A timeout occurred while attempting to restart either DQF or SAF processing. May be caused by a slow network link or large traffic volume.

User Action: Contact BEA MessageQ support if the problem does not correct itself.

MRS_E_SAF_ALLOC, Error allocating space for SAF list

Explanation: Fatal.

An attempt to allocate virtual memory by the MRS Server failed. This failure prevented the MRS Server from starting.

User Action: Increase the MRS servers page file quota in DMQ\$SET_SERVER_QUOTAS.COM.

MRS_E_SAFBYPASS, I/O Error caused beeps of SAF file: *filename*

Explanation: Error.

An error occurred while the MRS Server was processing a SAF file. The file was renamed with the extension `.SERR` and bypassed.

User Action: Check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFCLS, Error closing SAF file *filename*

Explanation: Error

An error occurred while MRS was closing a SAF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFDEL, Error deleting SAF file *filename*, status=*n*

Explanation: Error.

An error occurred while MRS was attempting to delete an empty SAF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFD_S, Error broadcasting SAFD_START for *b.g*

Explanation: Error.

The MRS Server failed to broadcast the SAFD_START message to the specified group.

User Action: None. The MRS Server will retry again later.

MRS_E_SAFERR, Unexpected error in SAF file *filename*

Explanation: Error.

MessageQ encountered a severe error when reading the specified SAF file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAF_FILES, SAF_FILES quota exceeded

Explanation: Error

An error occurred while MRS was attempting to create a new SAF file. The maximum number of SAF files has been reached.

User Action: Increase NUM_SAF_AREAS in group initialization file or increase AREA_SIZE to decrease the number of files required.

MRS_E_SAFIDXERR, Unexpected error building SAF list

Explanation: Error

An error occurred while attempting to access the Process List (PL).

User Action: Contact BEA MessageQ support.

MRS_E_SAFIODEL, IO error prevents use of SAF file filename

Explanation: Error.

An error occurred while MRS was attempting a file operation on the SAF file. The file has been renamed with extension .SERR.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFMSGDEL, Error Deleting message from SAF file filename

Explanation: Error.

MessageQ detected an error while deleting a message from the SAF.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFMSGTOOBIG, SAF message is too large for SAF file filename

Explanation: Error.

Message is too large to fit into the specified SAF file.

User Action: Increase the AREA_SIZE in the MRS section of the group initialization file.

MRS_E_SAFOPN, Error opening SAF file filename, status= n

Explanation: Error.

MessageQ detected an error when opening a SAF file.

User Action: Check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFRD, Error reading SAF file filename

Explanation: Error.

MessageQ detected an error when reading a SAF file.

User Action: Check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SAFTRUNC, Incomplete SAF record detected filename

Explanation: Error.

MessageQ has detected an incomplete record in the SAF file.

User Action: Check file system integrity.

MRS_E_SAFWR, Error writing SAF file filename

Explanation: Error.

MessageQ detected an error when writing to the SAF file.

User Action: Check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_SLIST, Unexpected error in list handling

Explanation: Error.

MessageQ detected an unexpected error.

User Action: Check page file quotas and contact BEA MessageQ support.

MRS_E_SPR_S, Error posting PAMS timer

Explanation: Error.

The MRS received an error status return code on a call to `pams_set_timer`.

User Action: Contact BEA MessageQ support.

MRS_E_SYNCCREATE, Error creating Journal file filename, status=n

Explanation: Error.

An error occurred while attempting to create a Journal file.

User Action: Depending upon the text of message, check disk quotas, open file limits, file system integrity, file protections, and ownership.

MRS_E_TIMERFAIL, Error occurred while queuing a timer

Explanation: Error.

MessageQ detected an error when queuing a timer.

User Action: Increase the ASTLM or TQELM quota for the MRS Server process in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_TL_ALLOC, Error allocating space for TL list

Explanation: Fatal.

An attempt to allocate virtual memory by the MRS Server failed. This failure prevented the MRS Server from starting.

User Action: Increase the MRS servers page file quota in `DMQ$SET_SERVER_QUOTAS.COM`.

MRS_E_TMOERR, Unexpected condition in timeout handling

Explanation: Error.

An unexpected error occurred in the MRS timer handling services.

User Action: Contact BEA MessageQ support.

MRS_E_UNKMSG, Received an Unknown MRS msg

+ Discarded msg - Src=*b.g* Tgt=*b.g* Class=*n* Type=*n* + Org-Src=*b.g* Org-Tgt=*b.g*
Size=*n* Seq=*x:x*

Explanation: Error.

An unexpected message was received by the MRS Server, which usually indicates an application program error. A program might have attempted a dialogue with the MRS Server.

User Action: Determine the source of the message and modify the application program.

MRS_I_DQFFILE, file:filename start:start_record end:end_record

Explanation: Informational.

DQF file information which usually coincides with an MRS error message.

User Action: None.

MRS_I_DQFSIZE, Max DQF areas:n, maximum size:n pages

Explanation: Informational.

Reflects the AREA_SIZE and NUM_DQF_FILES in pages.

User Action: None.

MRS_I_FREEENGOFF, Shutting off FREE file engine

Explanation: Informational.

All DMQ\$MRS: devices are full or there are no more I/O channels available. Shutting off FREE file engine.

User Action: Free disk space on DMQ\$MRS devices, or add another device to the DMQ\$MRS list, or increase system channel count, or drain the existing journal files to free space.

MRS_I_FREEENGOFF, Restarting FREE file engine

Explanation: Informational.

The FREE file engine is being restarted by the MRS server.

User Action: None.

MRS_I_GRP_NONE, Assuming default 1.0 recovery protocol for group:g

Explanation: Informational.

Recovery protocol for group is not known. Assuming 1.0 recovery protocol.

User Action: None.

MRS_I_INFO, information_text

Explanation: Informational.

MRS tracing information.

User Action: None.

MRS_I_JRNCLOSED, s journaling disabled for queue (b.g)

Explanation: Informational.

SAF or DQF journaling was disabled by either a SAF_SET_CLOSE or a DQF_SET_CLOSE request.

User Action: None.

MRS_I_JRNDISBYLOG, s journaling disabled by logical *logical_name*

Explanation: Informational.

User Action: None.

MRS_I_JRNDISBYMSG, s journaling disabled by message *message_type*, from (b.g)

Explanation: Informational.

User Action: None.

MRS_I_JRNENABYMSG, s journaling enabled by message *message_type*, from (b.g)

Explanation: Informational.

User Action: None.

MRS_I_JRNOOPENED, s journaling enabled for queue (b.g)

Explanation: Informational.

SAF or DQF journaling was enabled by either a SAF_SET_OPEN or a DQF_SET_OPEN request.

User Action: None.

MRS_I_JRNSETREFUSED, *message_type* message from (b.g) is refused

Explanation: Informational.

A journal control message was refused.

User Action: None.

MRS_I_ML_EXPAND, NUM_MESSAGES expanded to *n* entries

Explanation: Informational.

The message list, called the ML, has been expanded in order to handle a peak load that is greater than its current size.

User Action: Although this is an informational message, a user may wish to set the initial message list size large enough to store peak requirements and for efficient MRS Server operation. This can be controlled by the NUM_MESSAGES parameter in the group initialization file (DMQ\$INIT.TXT). Set it to at least the peak value reported by log event.

MRS_I_MLSIZE, Space allotted for *n* queues, *n* messages

Explanation: Informational.

MRS Server is reporting the allocated message list size which is based on the number of queues and number of messages as specified in the group initialization file.

User Action: None.

MRS_I_PL_EXPAND, NUM_QUEUES expanded to *n* entries

Explanation: Informational.

The queue list, called the PL, has been expanded in order to handle a peak load that is greater than its current size.

User Action: Although this is an informational message, a user may wish to set the initial queue list size large enough to store peak requirements and for efficient MRS Server operation. This can be controlled by the NUM_QUEUES parameter in the group initialization file (DMQ\$INIT.TXT). Set it to at least the peak value reported by log event.

MRS_I_RESTART, Group:*g* Placed on restart list from state=*server_state*

Explanation: Informational.

User Action: None.

MRS_I_RESTARTING, Attempting to restart group:*g*

Explanation: Informational.

The MRS server is attempting a handshake with the specified group's MRS server.

User Action: None.

MRS_I_RESTART_OFF, Removing group:*g* from restart list to state=*server_state*

Explanation: Informational.

User Action: None.

MRS_I_RP, Group:*g* Using recovery protocol:*n*

Explanation: Informational.

User Action: None.

MRS_I_RP_NONE, Assuming default 1.0 recovery protocol for msg (*x,x*)

Explanation: Informational.

Recovery protocol for message is not known. Assuming 1.0 recovery protocol.

User Action: None.

MRS_I_SAFFILE, file:filename start:start_record end:end_record

Explanation: Informational.

SAF file information which usually coincides with an MRS error message.

User Action: None.

MRS_I_SAFSIZE, Max SAF areas:n, maximum size:n pages

Explanation: Informational.

Reflects the AREA_SIZE and NUM_SAF_FILES in pages.

User Action: None.

MRS_I_STACK, MRS Server initialized

Explanation: Informational.

User Action: None.

MRS_I_START, MRS Server version Starting

Explanation: Informational.

User Action: None.

MRS_I_TOPO_EXPAND, MRS expanded group topology to n groups

Explanation: Informational.

The group topology list, called the TOPO, has been expanded in order to handle a peak load that is greater than its current size (the default is 128 groups).

User Action: None.

MRS_I_XGRPJRNCtrl, XGRP_JRN_CTRL is set to NO

Explanation: Informational.

A journal control request message was received from a remote group and was rejected because XGRP_JRN_CTRL in the MRS section of the group initialization file is set to NO.

User Action: To allow journal control messages from remote groups set the XGRP_JRN_CTRL parameter in the MRS section of the group initialization file to YES.

SBS Error Log Messages

This section lists and describes SBS Server error log messages. Use the following Key to understand conventions used in this section:

Key:

- g = group ID decimal value
- q = queue number decimal value
- n = numeric value
- x = hexadecimal value
- s = string

E_IO_ERROR: on channel ($g.q$), iosb[0]:0X x , iosb[1]:0X x

Explanation: Error.

The SBS Server detected an Ethernet I/O error.

User Action: Check the hardware and network error counts on the Ethernet device.

E_IO_RESTART: on channel ($g.q$), restart code: n

Explanation: Informational.

The SBS Server is attempting to restart the protocol on an ethernet logical link after an automatic LAN shutdown.

User Action: None.

Ethernet Error:% s on channel:($g.q$)

Explanation: Error.

The SBS Server detected an Ethernet Error.

User Action: Check the hardware and network error counts on the Ethernet device.

Ethernet MOT Declaration n --- Received sequence gap on Channel n from Group g Received 0X x Expected 0X x

Explanation: Warning.

The queuing capacity of the system has been exceeded.

User Action: The sender program should slow the message send rate.

Ethernet received sequence gap on channel n from group n Received 0X x Expected 0X x

Explanation: Warning.

The SBS Server detected an Ethernet sequence gap.

User Action: The queuing capacity of the system has been exceeded. The sender program should slow the message send rate.

**Ethernet MOT Declaration *n* - Received sequence gap on channel *n* from group *g*
Received 0X*x*, expected 0X*x***

Explanation: Warning.

The SBS Server detected a sequence gap in the MOT declaration.

User Action: The queuing capacity of the system has been exceeded. The sender program should slow the message send rate.

Group record not found for group: *n*

Explanation: Warning.

The SBS Server cannot locate the group in the group table.

User Action: Contact BEA MessageQ support.

***** Failed to initialize SBS Server *****

Explanation: Fatal.

The SBS Server was not able to initialize.

User Action: Check the SBS Server and EVL log files for related errors. Also check the SBS Server's page file quotas.

Failure to locate Registration AVAIL target_*q* '*g.q*' Distribution_*Q* '*g.q*'

Explanation: Warning.

The SBS Server cannot locate the specified AVAIL registration in its database.

User Action: Modify the sender program to use the correct information in the AVAIL_DEREG message.

Failure to locate Registration in SBS_DEREG_BY_ID reg_id = '*n*' source = '*g.q*'

Explanation: Warning.

The SBS Server cannot locate the specified SBS registration ID in its database.

User Action: Modify the sender program to use the correct registration ID.

Failure to locate Registration SBS_DEREG MOT '*n*' dist_*q*:*g.q* source:*g.q*

Explanation: Warning.

The SBS Server cannot locate the specified SBS registration ID in its database.

User Action: Modify the sender program to use the correct registration ID.

Failure to locate Registration SBS_DEREGISTER MOT '*n*'

Distribution '*g.q*'

source = '*g.q*'

Explanation: Warning.

The SBS Server cannot locate the specified SBS registration ID in its

database.

User Action: Modify the sender program to use the correct registration ID.

Invalid CLASS = 'n' target = SBS_SERVER type = 'n' source = 'g.q'

Explanation: Warning.

Usually occurs when the application program attempts a dialogue with SBS Server and uses the incorrect message class field.

User Action: Modify the sender application program.

Invalid distribution in MOT_REG_EZ

nlow = 'g.q'

nhigh = 'g.q'

nsource = 'g.q'

Explanation: Warning.

The SBS server detected an invalid distribution address.

User Action: Correct the application code.

Invalid Ethernet Address Specified 's' for channel = 'n'

Explanation: Fatal.

The SBS Server detected an invalid Ethernet address.

User Action: Edit the group initialization file and correct the multicast assignments SBS section.

Invalid Ethernet Protocol Specified 's' for channel = 'n'

Explanation: Fatal.

The SBS Server detected an invalid Ethernet control channel protocol.

User Action: Edit the group initialization file and correct the control channel protocol assignments in SBS section.

Invalid ethernet ucb start msg length

Explanation: Error.

The SBS Server detected an invalid Ethernet UCB control message.

User Action: Contact BEA MessageQ support.

Invalid group in distribution in MOT_REG_EZ

nlow = 0xx

nhigh = 0xx

nsource = 'b.g'

Explanation: Warning.

The SBS server detected an invalid distribution group.

User Action: Correct the application code.

Invalid MOT Address = 'n' from 'g.q'

Explanation: Warning.

The application program used an invalid Multipoint Outbound Target (MOT) address.

User Action: Modify the sender program to use a broadcast address within the legal MOT range.

Invalid MOT Address range request from 'g.q'

Explanation: Warning.

The SBS Server detected an invalid MOT range in an SBSE_STATUS_REQ message.

User Action: Modify the application program to use a valid MOT range.

Invalid MOT Range in MOT_REG_EZ

low = 'n'

high = 'n'

source = 'g.q'

Explanation: Warning.

The SBS Server detected an invalid MOT range in a MOT_REG_EZ message.

User Action: Modify the sender program to use a broadcast address within the legal MOT range.

Invalid mot_count:%d in SBS_CS_IMSG command. LOAD ignored

Explanation: Warning.

The SBS Server detected an invalid mot count in an SBS_CS_IMSG LOAD command.

User Action: Contact BEA MessageQ support.

Invalid registration in SBS_DEREG_BY_ID

reg_id = 'n' source = 'g.q'

Explanation: Warning.

The SBS Server cannot locate the specified registration ID in its database.

User Action: Modify the sender program to use the correct registration ID.

Invalid version number in message_type message

version = 'n'

source = 'g.q'

Explanation: Warning.

The SBS Server detected an invalid message version number in the specified message.

User Action: Modify the sender program to use the correct message version number.

Invalid version number n in *message_type*

Explanation: Warning.

The SBS Server detected an invalid version number in the specified message.

User Action: Modify the application program to use the correct message version number.

Invalid version number in Event Registration Msg $ver=n$ $tgt=g.q$

Explanation: Warning.

The SBS Server detected an invalid message version number in the specified message.

User Action: Modify the sender program to use the correct message version number.

LINK_COMPLETE msg from $g.q$ has invalid group g , largest group is g - Message Ignored

Explanation: Warning.

SBS Server detected a message with an LINK_COMPLETE message with for invalid group number.

User Action: Contact BEA MessageQ support.

LINK_LOST message, from $g.q$, with invalid group g , largest group is g - Message Ignored

Explanation: Warning.

The SBS Server detected a message with a LINK_LOST message with for an invalid group number.

User Action: Contact BEA MessageQ support.

MOT Declaration n --- Received sequence gap from group g Received $0Xx$, expected $0Xx$

Explanation: Warning.

The SBS Server detected a sequence gap in the MOT declaration.

User Action: The queuing capacity of the system has been exceeded. The sender program should slow the message send rate.

PAMS_FREE_LMSG Failed

Explanation: Warning.

The SBS Server failed to free an internal message buffer.

User Action: Check page file quotas and other system parameters. Look for other errors in the EVL or SBS log that may give a better indication of the source of the problem.

PAMS__MSGUNDEL to group 'g'

Explanation: Warning.

A message was returned to the SBS server because it could not be delivered to the target queue.

User Action: Check cross group links to the specified group and check to see if the SBS Server is still running on specified group.

PAMS_RCV_LMSGW Failed in Main loop

Explanation: Warning or Fatal.

The SBS Server received an error status when attempting to read a message from the SBS Server queue. If the error status was PAMS__BADLOGIC or PAMS__PAMSDOWN then the error is Fatal and the SBS Server will shutdown; otherwise, SBS Server will keep trying.

User Action: Check server and EVL log files for source of problem. If the problem cannot be determined then contact BEA MessageQ support.

Received out of range DNA mot:n from:(g,q)

Explanation: Warning.

The SBS Server detected a DNA MOT that was out of range.

User Action: Modify the sender program or increase the MOT range in the group initialization file.

Rexmit request to group g [MOT n; ;seq n;length n;hdr:n]

Explanation: Warning.

A MSG_TYPE_SBSE_REXMIT_REQ message was returned to the SBS server because it could not be delivered to the target groups SBS server.

User Action: Check cross group links to the specified group and check to see if the SBS Server is still running on specified group.

SBS Ethernet MOT 'n' BROADCAST rejected --- no channel ---

Explanation: Error.

The MOT is not assigned to the Ethernet channel and DMQ\$ETH_MOT_ONLY logical name = YES

User Action: Add the correct MOT definitions to the the group initialization file.

SBS Ethernet MOT 'n' no Ethernet channel assigned

Explanation: Warning.

MOT not assigned to the Ethernet channel.

User Action: Add the correct MOT definitions to the the group initialization file.

SBS Ethernet MOT 'n' REGISTRATION refused --- no channel ---

Explanation: Warning.

MOT not assigned to the Ethernet channel and `DMQ$ETH_MOT_ONLY` logical name = YES

User Action: Add the correct MOT definitions to the the group initialization file.

SBS received message with invalid group 'g' largest group:g source:g.q - Message Ignored

Explanation: Warning.

SBS Server detected a message with an invalid source group number.

User Action: Contact BEA MessageQ support.

SBS Server Initialized

Explanation: Informational.

User Action: None.

SBS Server (version.endian) Starting

Explanation: Informational.

User Action: None.

**SBS Received sequence gap from group g
Received 0Xx, expected 0Xx**

Explanation: Warning.

The SBS Server detected a sequence gap.

User Action: The queuing capacity of the system has been exceeded. The sender program should slow the message send rate.

**SBS_INIT_ACK received message with invalid group 'g' largest group is 'g'
source = 'g.q'**

Explanation: Warning.

SBS Server detected a message with an `SBS_INIT_ACK` message with an invalid source group number.

User Action: Contact BEA MessageQ support.

**SBS_INIT_ACK_E received message with invalid group 'g' largest group is 'g'
source = 'g.q'**

Explanation: Warning.

SBS Server detected a message with an `SBS_INIT_ACK_E` message with an invalid source group number.

User Action: Contact BEA MessageQ support.

SBS_INIT_E received message from *g.g* with invalid group *g*

Explanation: Warning.

SBS Server detected an SBS_INIT_E message with an invalid group number.

User Action: Contact BEA MessageQ support.

Index

A

- access control list (ACL) 13-4
 - setting 13-5
- and 10-1
- API routines
 - testing 9-6
- application startup procedure 2-23
- attaching to a group's logical name table 2-24

B

- buffer pool configuration 9-4
- buffer pool configuration table 2-16, 4-8
 - parameters 4-9
- bus ID
 - creating 2-5
 - reserved 2-6
- bus numbers
 - changing 9-10

C

- class codes 7-12
- Cleint Library Server
 - supported transports 8-3
- Client Library Server 8-1
 - event logging 8-15
 - files installed 8-3
 - linking 8-3
 - overview 8-2
 - security 8-8

- Single-Client mode 8-7
 - starting with DCL 8-13
 - starting with Manager utility 8-12
 - stopping with Manager utility 8-13
 - tracing 8-15
- Client Library Server configuration table 2-16
- CLS
 - See Client Library Server 8-1
- COM Server 10-6
 - shutdown command 9-10
 - shutting down 10-7
 - starting and shutting down 9-3
- COM Server counters
 - resetting 10-7
- communications 9-4
- configuration
 - editing DMQ\$INIT.TXT 2-10
 - initial steps 2-3
- configuration files
 - sharing 3-12
- cross-group connection table 2-15
 - fields 3-3
 - overview 3-3
 - transports 3-6
- cross-group connections 3-1
 - re-establishing 10-9
- CUSTOMIZE command procedure 9-6

D

- Dead Letter Journal (DLJ) 5-2
- DEC/UCX link driver 3-11
- DECnet intrusion alarms
 - suppressing 3-13
- DECnet transports 3-6
- DEFAULT_NAMESPACE_PATH
 - parameter 7-4
- Destination Queue file (DQF) 5-2
- detaching a process
 - with DCL context 2-24
 - without DCL context 2-25
- Distributed Name Service (DNS) 7-7
- DMQ\$BOOT.COM
 - editing with CUSTOMIZE procedure 9-6
- DMQ\$COM_SERVER process
 - starting and shutting down 9-3
- DMQ\$CREATE_GROUP.COM 2-6
- DMQ\$DEBUG logical 10-12
- DMQ\$DELETE_GROUP 2-31
- DMQ\$GMT_OFFSET 3-15
- DMQ\$GROUP_OUTPUT logical 10-14
- DMQ\$INIT.TXT 2-10
 - buffer pool configuration table 2-16, 4-8
 - Client Library Server configuration table 2-16
 - cross-group connection table 2-15
 - editing with CUSTOMIZE procedure 9-6
 - Message Recovery System server section 2-19
 - message routing table 3-10
 - Naming Agent section 2-20
 - parameters modifiable at run-time 2-29
 - sections 2-11
 - Selective Broadcast Server section 2-18
- DMQ\$INIT.TXT file 9-9
- DMQ\$LLS_VERIFY utility 9-4
- DMQ\$LOADER
 - for loading run-time changes 2-29
- DMQ\$LOADER Utility 9-9
- DMQ\$LOOP 9-5
- DMQ\$LOOP utility 9-4
- DMQ\$MGR_UTILITY 10-1
- DMQ\$MONITOR utility 10-1
- DMQ\$PROCESS_OUTPUT logical 10-14
- DMQ\$PROCESS_START 2-24
- DMQ\$SERVER_TRACE logical 10-12
- DMQ\$SET_LNM_TABLE.COM 2-8
- DMQ\$SHUTDOWN.COM 2-26
- DMQ\$STARTUP procedure 9-8
- DMQ\$STARTUP.COM
 - running without starting servers 2-22
 - starting multiple versions of BEA MessageQ 2-23
- DMQ\$TEST utility 9-6
- DMQ\$TRACE_OUTPUT logical 10-12
- DMQ\$TYPCLS.TXT 7-13
- DMQCLSEC.TXT 8-8
- DMQNS_DEFAULTPATH logical 7-4
- DMQNS_DEFAULTPATH logical name 2-15

DNS

- See Distributed Name Service 7-7
- dual-rail mode 6-12
- During 8-3

E

- editor
 - default 2-32
- ENABLE_SBS parameter 6-2
- error logging
 - redirecting 10-12
- Ethernet
 - retransmission protocol 6-11
- Ethernet Broadcasting parameters 6-9
- Ethernet multicasting 6-1
 - maximum message size 6-2
- event log file 10-14

- Event Logger process (EVL_LOG) 10-14
- Event Logging 2-28
- event logging
 - dynamically redirecting 10-14

G

- global memory
 - configuring 4-1
 - sections 4-7
- group
 - creating 2-6
 - deleting 2-31
 - logical name table 2-8
- group details
 - displaying 10-9
- group ID 2-6
- group name table
 - displaying entries 10-15
- group numbers
 - changing 9-10
- groups 9-10

I

- initialization file
 - naming section 7-2
- Installation Verification Procedure (IVP) 9-3

J

- Journal Controls menu 10-16
- journal files
 - using auxiliary files 5-3
- journal management 10-16
- journals
 - managing DLJ and PCJ 10-17
 - managing DQF and SAF 10-16
 - retrieving messages 5-5

L

- link connect table 10-9
- link details
 - displaying 10-8
- link drivers
 - TCP/IP 3-11
- link summary 10-8
- LOADER utility 9-9
 - restrictions 9-9
- log files
 - redirecting 2-33
- logical name table 2-8

M

- main menu 9-1
 - invoking 9-2
- Manager utility 10-1
- message exchange methods 3-2
- message queueing group
 - resources 2-5
- message queues
 - configuring 4-1
- message recovery
 - configuring 5-1
- message recovery journals 5-2
- message recovery services
 - capabilities 5-2
- Message Recovery Services (MRS) 5-1
- message recovery system
 - auxiliary journals 5-2
- Message Recovery System server section 2-19
- message routing 3-8
 - dynamic 3-8
 - static 3-8
- message routing table 3-8, 3-10
- monitor 10-1
- Monitor utility 10-1, 10-4
- monitoring system status 10-1
- MOT_PRIVATE 6-4

MOT_RESERVED parameter 6-4
MOT_UNIVERSAL parameter 6-4
MultiNet link driver 3-11
Multipoint Outbound Target (MOT)
 specifying addresses 6-3

N

name
 fully qualified 7-6
 partially qualified 7-6
 unqualified 7-6
names
 displaying 10-19
 global dynamic 7-5
 global static 7-5
 managing address associations 10-21
 removing from namespace 10-22
 view cached 7-8
namespace 7-4
 DNS 7-2
 lightweight 7-2
 managing 7-8
namespace path
 configuring default 7-4
namespace path definition
 default 2-15
Naming 10-18
naming 7-1
 examples 7-9
Naming Agent 2-28
Naming Agent menu 10-18
Naming Agent section 2-20
Naming Agent Server 7-2

O

optimized Ethernet mode
 configuring 6-11

P

post confirmation journal (PCJ)
 using 5-5
Post Confirmation Journal (PCJ) 5-2
PRIVATE range 6-4
privileges
 system management account 2-2
process
 forcing exit 10-11
process event 10-14
profile section of DMQ\$INIT.TXT 2-12

Q

queue configuration table 2-17
 parameters 4-3
 rules 4-5
queue counters
 displaying 10-5
queue details
 displaying 10-10
queue naming rules 4-5
queue numbers
 permanent 4-5
 temporary 4-5
queue quotas
 displaying 10-5
queue status
 displaying 10-7
queue summary
 displaying 10-10
queues
 attaching to undefined 8-11
 flushing 10-10
 special configuration issues 8-9
 starting 10-11
 stopping 10-11

R

receive silo 6-13

- remote groups
 - monitoring 10-9
- restrictions 9-9
- retransmission protocol
 - Ethernet 6-11
- Retransmission Protocol (RP)
 - Ethernet multicasting 6-1
- routing table
 - displaying 10-7
- RP/ETH Configuration 6-16
- RP/ETH Retransmission Protocol 6-12
- run-time parameter modification 2-29

S

- SBS Server
 - configuring 6-3
 - dual-rail mode Ethernet 6-12
 - interoperability 6-10
 - restrictions 6-9
- SBS server
 - optimized Ethernet mode 6-4
- SBS Server parameters 6-6
- security file 8-8
- Selective Broadcast Server 2-18
- Selective Broadcast Services
 - COMM_SERVICE 6-2
 - version compatibility 6-2
- Selective Broadcast Services (SBS) 6-1
- shut down a BEA MessageQ group 2-26
- shutting down BEA MessageQ 9-10
- starting BEA MessageQ 2-21
- startup
 - synchronizing 2-33
- status logging 10-12
- Store and Forward file (SAF) 5-2
- support
 - technical xix
- symbols
 - defining 2-32
- system management 10-1

- system management account
 - privileges 2-2

T

- TCP/IP link drivers 3-11
- TCPware link driver 3-11
- time zone adjustments 3-15
- trace output targets 10-12
- tracing
 - enabling 10-12
 - enabling dynamically 10-13
- type codes 7-12

U

- UNIVERSAL range 6-4
- utilities
 - DMQ\$CREATE_GROUP.COM 2-6
 - DMQ\$SET_LNM_TABLE.COM 2-8