BEA MessageQ

Client for OpenVMS User's Guide

BEA MessageQ for OpenVMS Version 5.0
Document Edition 2.0
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 Client for OpenVMS User's Guide

Document Edition	Date	Software Version
2.0	March 2000	BEA MessageQ Client for OpenVMS, Version 5.0

Contents

3. Configuring the BEA MessageQ Client

4. Using the BEA MessageQ Client

5. Troubleshooting

Index

Preface

Purpose of This Document

This document provides an introduction to the BEA MessageQ Client for OpenVMS and provides instructions on installing, configuring, using, and troubleshooting the BEA MessageQ Client for OpenVMS software.

Who Should Read This Document

This document is intended for the following audiences:

- system installers who will install BEA MessageQ on supported platforms
- system administrators who will configure, manage, and troubleshoot BEA MessageQ on supported platforms
- applications designers and developers who are interested in designing, developing, building, and running BEA MessageQ applications

How This Document Is Organized

BEA MessageQ Client for OpenVMS User's Guide is organized as follows:

- Chapter 1, "Introduction" describes the capabilities, benefits, and architecture of the BEA MessageQ Client for OpenVMS.

- Chapter 2, “Installing the BEA MessageQ Client” describes how to install the BEA MessageQ Client for OpenVMS software, including prerequisites, installation procedures, recovering from installation errors, and post-installation tasks.
- Chapter 3, “Configuring the BEA MessageQ Client” describes how to configure the MessageQ Client server connection, message recovery services, and logging and tracing capabilities. It also includes instructions on testing the configuration.
- Chapter 4, “Using the BEA MessageQ Client” describes how to develop, run, and manage MessageQ Client applications.
- Chapter 5, “Troubleshooting” describes how to identify and correct problems while running your MessageQ client applications.

How to Use This Document

This document 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 access it as an online document via the BEA MessageQ Online Documentation CD. 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 `index.htm` file in the top-level directory of the BEA MessageQ Online Documentation CD. On the main menu, click the Introduction to Message Queuing button.

Note: The online documentation requires a Web browser that supports HTML version 3.0. Netscape Navigator version 3.0 or later, or Microsoft Internet Explorer version 3.0 or later are 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. If your browser offers a Print Frames feature, you can use the feature to select the frame containing the chapter or appendix you want 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. On the CD's main menu, click the Bookshelf button. On the Bookshelf, scroll to the entry for the BEA MessageQ document you want to print and click the PDF option.

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

`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.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON 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.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item

. . .	Indicates one of the following in a command line:
	♦ 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.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	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 software, related BEA publications, and other publications related to the technology.

BEA MessageQ Documentation

The BEA MessageQ information set consists of the following documents:

BEA MessageQ Introduction to Message Queuing

BEA MessageQ Installation and Configuration Guide for Windows NT

BEA MessageQ Installation and Configuration Guide for UNIX

BEA MessageQ Installation Guide for OpenVMS

BEA MessageQ Configuration Guide for OpenVMS

BEA MessageQ Programmer's Guide

BEA MessageQ FML Programmer's Guide

BEA MessageQ Reference Manual

BEA MessageQ System Messages

BEA MessageQ Client for Windows User's Guide

BEA MessageQ Client for UNIX User's Guide

BEA MessageQ Client for OpenVMS 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 ProductName, or if you have problems installing and running ProductName, 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 Introduction

This section provides an introduction to the BEA MessageQ Client and covers the following topics:

- What is the BEA MessageQ Client?
- Benefits of Using the BEA MessageQ Client
- Architectural Overview

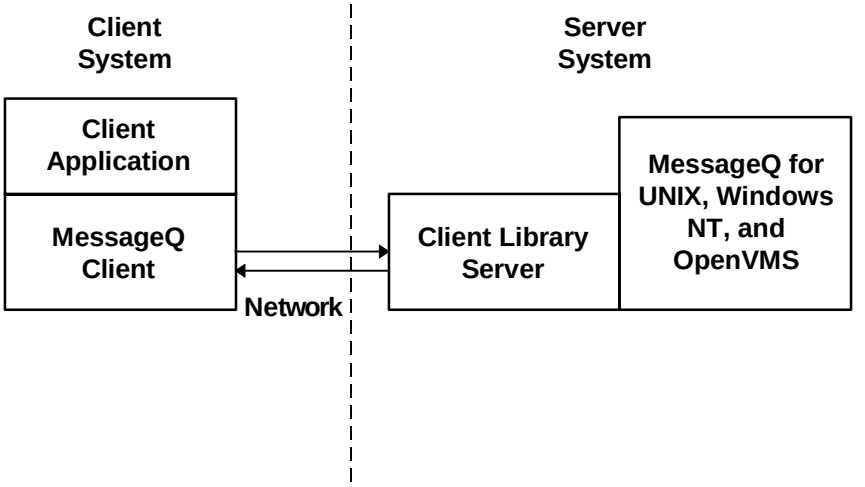
What is the BEA MessageQ Client?

The BEA MessageQ Client is a client implementation of the BEA MessageQ Application Programming Interface (API). It provides message queuing support for distributed network applications using a BEA MessageQ Server to provide reliable message queuing for distributed multi-platform network applications. The BEA MessageQ Client is referred to as a "light-weight" implementation of BEA MessageQ because it requires fewer system resources (disk space and memory) and less configuration and management than a BEA MessageQ Server.

The BEA MessageQ Client is connected to the message queuing bus through a network connection with a Client Library Server (CLS) on a remote BEA MessageQ Server. The CLS acts as a remote agent to perform message queuing operations on behalf of the BEA MessageQ Client. The CLS runs as a background server to handle multiple BEA MessageQ Client connections. The BEA MessageQ Client establishes a network connection to the CLS when an application attaches to the message queuing bus. The CLS performs all communication with the client application until the application detaches from the message queuing bus. The network connection to the CLS is closed when the application detaches from the message queuing bus.

BEA MessageQ Clients are available for Windows 95, Windows NT, most popular UNIX systems, OpenVMS, and IBM MVS systems. See Figure 1-1 for a diagram of the BEA MessageQ Client and Server components.

Figure 1-1 BEA MessageQ Client and Server Components



The BEA MessageQ Client allows multiple applications to connect to separate queues on the message queuing bus. A separate network connection to the CLS is maintained for each BEA MessageQ Client application. The message queuing operations and network activities of each client are isolated from other clients. The total number of applications that can connect to the message queuing bus is limited by the number of TCP/IP or DECnet sessions. To provide robust network connections, a backup CLS can be configured for automatic failover if the primary CLS becomes unavailable. The BEA MessageQ Client can also automatically reconnect to the CLS following a network failure.

When running the BEA MessageQ Client on Windows systems, users can select the standard request/response protocol or the Minimum Packet Protocol (MPP) to reduce network traffic. There is no performance advantage by using MPP on high-speed local area networks, because MPP is designed for use in wireless networks or dialup connections with network speeds lower than local area networks.

When the connection to the CLS is unavailable, the BEA MessageQ Client provides recoverable messaging using a local store-and-forward (SAF) journal to store recoverable messages. When the connection to the CLS is reestablished, all messages in the SAF journal are sent before new messages are processed.

The BEA MessageQ Client for Windows supports a variety of popular application development environments and languages including C/C++, Visual Basic, PowerBuilder, and others. In addition, it includes several utility programs to monitor and test applications. The BEA MessageQ clients for UNIX and OpenVMS systems provide language bindings for C and C++. Example programs demonstrate the use of various features of the BEA MessageQ API.

Benefits of Using the BEA MessageQ Client

The BEA MessageQ Client provides the following benefits:

- Reduces system resource load
- Reduces system management overhead
- Provides network protocol independence

The BEA MessageQ Client provides message queuing capabilities for BEA MessageQ applications using fewer system resources (shared memory and semaphores) and running fewer processes than a BEA MessageQ Server. Therefore, the BEA MessageQ Client enables distributed BEA MessageQ applications to run on smaller, less powerful systems than the systems required to run a BEA MessageQ Server.

Run-time configuration of the BEA MessageQ Client is extremely simple. A minimal configuration requires only the name of the server system, the network endpoint to be used by the CLS, and the desired network transport. Running the BEA MessageQ Client makes it unnecessary to install and configure a BEA MessageQ Server on each system in the network. Instead, a distributed BEA MessageQ environment can consist of a single system running a BEA MessageQ UNIX, Windows NT, or OpenVMS Server and one or more systems running BEA MessageQ Clients.

For example, suppose a small business has 10 networked workstations that need to run a BEA MessageQ application. Under previous versions of BEA MessageQ, it would be necessary to install, configure, and manage a message queuing group on each

workstation. Now, however, by using the BEA MessageQ Client, a BEA MessageQ Server need be installed and configured only on a single workstation. Installing the BEA MessageQ Client on the remaining nine workstations provides message queuing support for all other BEA MessageQ applications in the distributed network.

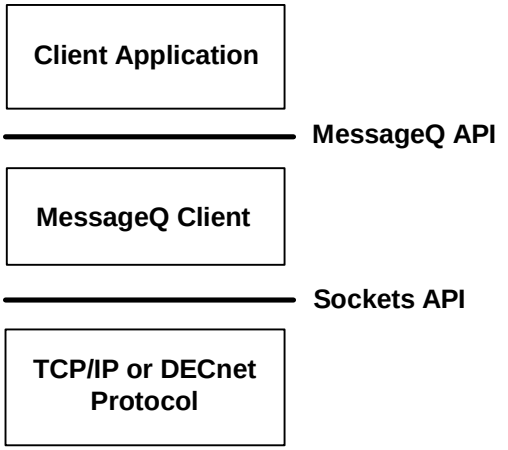
In this example, only one workstation needs to be sized and configured to optimize performance, reducing the burden of system management to a single machine. System management and configuration for the remaining systems is drastically simplified because managing the BEA MessageQ Client consists mainly of identifying the BEA MessageQ Server that provides full message queuing support. The BEA MessageQ Client can be reconfigured quickly and easily and multiple clients can share the same configuration settings to further reduce system management overhead.

The BEA MessageQ Client performs all network operations for client applications, making it unnecessary for a client program to be concerned about the underlying network protocol. The BEA MessageQ Client enhances the portability of applications, enabling them to be ported to a different operating system and network environment supported by BEA MessageQ with no change to the application code.

Architectural Overview

The BEA MessageQ Client for OpenVMS provides a shareable image supporting multiple BEA MessageQ-enabled applications. An object library is also provided for applications that need to use static linking. The BEA MessageQ Client is available for both VAX and Alpha processors. Figure 1-2 shows the BEA MessageQ Client for OpenVMS architecture.

Figure 1-2 BEA MessageQ Client for OpenVMS Architecture



The BEA MessageQ Client allows multiple applications to connect to separate queues on the message queuing bus. A separate network connection to the CLS is created for each client application. The total number of applications that can connect to the message queuing bus is limited by the number of TCP/IP or DECnet sessions. On Windows systems, the Client DLL uses the Windows Sockets API for network services.

If the network connection to the CLS is lost or unavailable, the BEA MessageQ Client optionally stores messages in a local journal file for later retransmission.

The Client Library Server

The Client Library Server (CLS) is a BEA MessageQ application that runs as a background server. The CLS performs all communication with the BEA MessageQ Client for each client application until the application detaches from the message queuing bus. The message queuing operations and network activity of each client are isolated from other clients.

The CLS supports multiple client connections using the following techniques:

- On Windows NT Server systems, the CLS is multi-threaded.

- On UNIX Server systems, the CLS uses a separate process to handle each BEA MessageQ Client connection.
- On OpenVMS Server systems, the CLS can operate in two modes:

Single-client mode	A separate CLS process is created to support each remote client.
Multi-client mode	A single CLS process supports multiple clients using asynchronous message queuing operations provided by BEA MessageQ for OpenVMS systems.

When the CLS starts, it initializes a listener process (or thread) that establishes a network endpoint and waits for connections from a client application. The endpoint on which the CLS listens is determined by the command-line arguments used to start the CLS.

BEA MessageQ Client applications attempt to connect to the CLS when they initiate an attach queue operation. The BEA MessageQ Client uses configuration information in the `dmq.ini` configuration file. The CLS creates a server subprocess (or server thread) for each new client connection. The server subprocess terminates when the client detaches from the bus, or the network connection is closed.

The CLS can use a security file (located on the server system) to control client access to the message bus. Client access can be restricted to specific queues or CLS endpoints.

How the BEA MessageQ Client and CLS Work Together

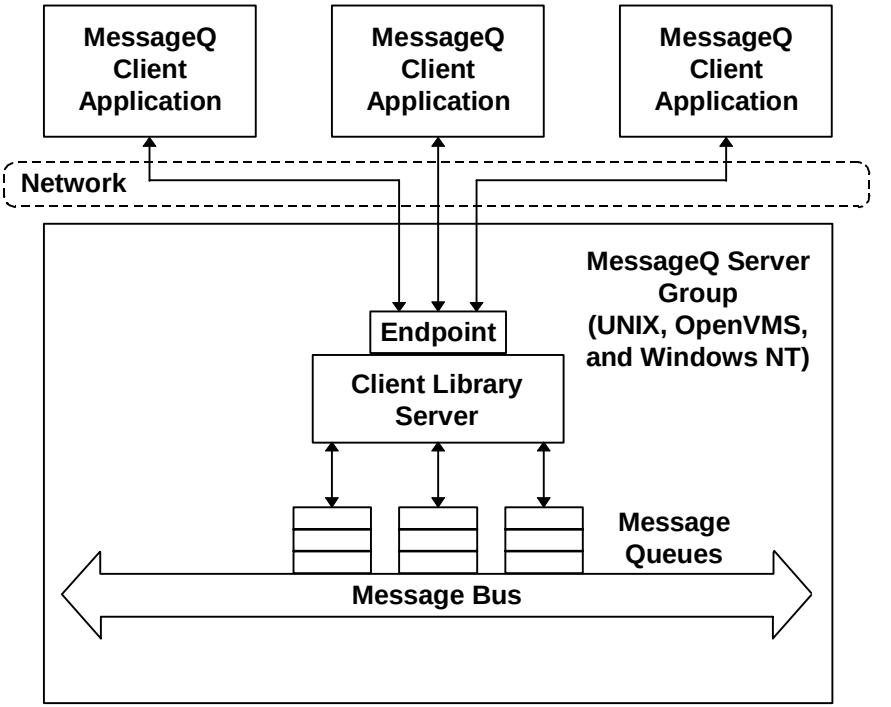
The BEA MessageQ Client uses a request/response protocol to communicate with a Client Library Server (CLS) running on a BEA MessageQ server system. The BEA MessageQ Client is called a light-weight client connection to the BEA MessageQ message queuing bus because it relies on a BEA MessageQ Server for the following:

- Message queues for all BEA MessageQ Client applications are implemented on a remote system running a BEA MessageQ Server group.
- Message delivery to target queues is provided by the Queuing Engine, a server process that runs on a BEA MessageQ Server.

- Message routing and cross-group transport among multiple BEA MessageQ Server systems and other BEA MessageQ Client applications are provided by the BEA MessageQ Server group.
- Guaranteed message delivery is provided by the MRS capability of the BEA MessageQ Server group. The BEA MessageQ Client provides a local store-and-forward (SAF) journal for temporarily storing recoverable messages when the connection to the CLS is not available.

Figure 1-3 shows the relationship of the BEA MessageQ Client and CLS to the BEA MessageQ message queuing bus.

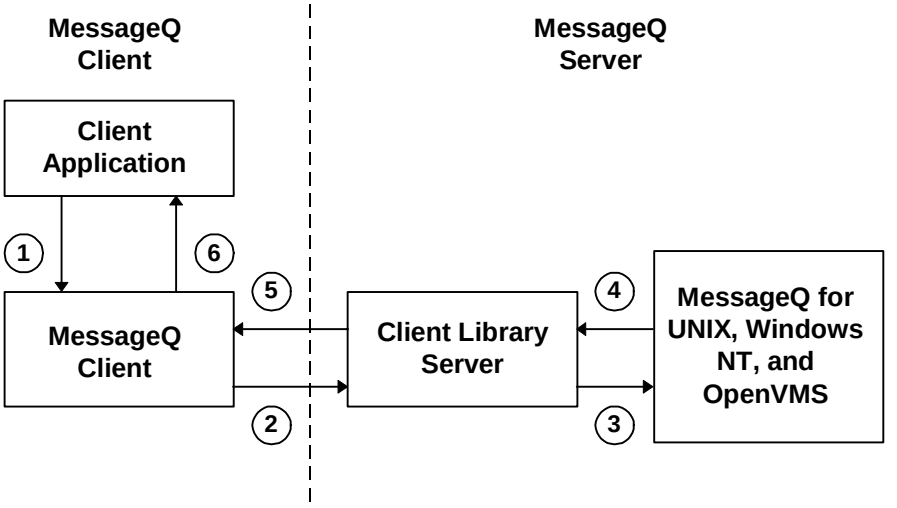
Figure 1-3 BEA MessageQ Client and CLS Architecture



All BEA MessageQ Client API functions supported by the CLS are processed using the following sequence of events as shown in Figure 1-4.

1. The client application makes a BEA MessageQ function call to the BEA MessageQ Client.
2. The BEA MessageQ Client verifies the function call arguments and sends them in a request to the CLS, which is waiting to receive client requests.
3. When a request arrives, the CLS makes the corresponding BEA MessageQ function call in the BEA MessageQ Server group.
4. The BEA MessageQ function completes, and returns the results to the CLS.
5. The CLS sends the return parameters and function status in a response back to the BEA MessageQ Client that initiated the request.
6. The BEA MessageQ Client function call returns to the application with the return arguments and function status.

Figure 1-4 How Client Application Requests are Processed



2 Installing the BEA MessageQ Client

This chapter describes how to install the BEA MessageQ Client for OpenVMS software. It includes the following topics:

- Installation Prerequisites
- Installing the BEA MessageQ Client Software on OpenVMS Systems
- Recovering from Errors During the Installation
- Postinstallation Tasks

Installation Prerequisites

To successfully install BEA MessageQ Client software on your client machine, you must ensure that your environment meets the following installation requirements:

- Hardware
- Software
- Disk space
- System disk backup

Hardware Requirements

To perform the installation, you need the following hardware:

- Compaq VAX or Alpha processor
- CD-ROM drive to read software distribution media
- Approximately 41K blocks of free disk space on the working device for installations on an Alpha system, or 61K blocks of free space for installation on a VAX system
- A PC (for obtaining the licence file from a diskette)

Software Requirements

Your environment must meet the following software requirements to run the BEA MessageQ Client software:

1. BEA MessageQ for OpenVMS, Windows NT, or UNIX must be installed and running on a server machine.

   If the BEA MessageQ software has not already been installed, see the Installation and Configuration Guide for your server platform to install BEA MessageQ software.

2. A minimum of 1 bus and 1 group must be configured on the BEA MessageQ server.

   If the BEA MessageQ software does not have a bus and group configured, see the Installation and Configuration Guide for your server platform to configure a BEA MessageQ bus and group.

3. TCP/IP or DECnet network software must be installed and running on both the client and server systems. (DECnet is only supported on OpenVMS systems.)

4. If you intend to develop client applications, you must have a program development environment that allows you to compile and link your applications.

Disk Space Requirements

BEA MessageQ Client systems require approximately 24K blocks of free disk space on Alpha systems and 51K blocks of free disk space on VAX systems to hold the BEA MessageQ Client installation files.

Backing Up Your System Disk

We recommend that you back up your system disk before installing any software. For details on performing a system disk backup, see the system documentation for your server platform.

Installing the BEA MessageQ Client Software on OpenVMS Systems

The BEA MessageQ Client installation dialog displays a list of options allowing you to choose which subsets to install. Table 2-1 describes the BEA MessageQ Client installation subsets.

Table 2-1 BEA MessageQ Client Installation Subsets

Installation Option	Description
Base Kit	BEA MessageQ Server software
Client Library Kit	An archive library that supports BEA MessageQ-enabled applications

Perform the following steps to install BEA MessageQ Client software on OpenVMS systems. You can stop the installation procedure at any time by pressing Ctrl/C. When you press Ctrl/C to stop the installation, files created up to that point are not automatically deleted. You must manually delete those files.

The steps to install the BEA MessageQ Client are as follows:

Step 1—Run the VMSINSTAL procedure

Log into the system account and run the VMSINSTAL procedure as indicated below to install the BEA MessageQ Client for OpenVMS. The installation procedure checks to make sure that you are satisfied with the backup of your system and that you have the correct product to install:

```
$ @sys$update:vmsinstal MQAXP050 DKA100:[KITS.DMQ050]
OpenVMS AXP Software Product Installation Procedure V6.2
It is  5-NOV-1999 at 14:35.
Enter a question mark (?) at any time for help.
%VMSINSTAL-W-ACTIVE, The following processes are still active:
       SYSTEM
* Do you want to continue anyway [NO]? y
* Are you satisfied with the backup of your system disk [YES]?
* Where will the distribution volumes be mounted: $111$DKC400:[MQVMSV50.KIT]
Enter the products to be processed from the first distribution volume set.
* Products: MQAXP050
* Enter installation options you wish to use (none):
The following products will be processed:
  MQAXP V5.0
Beginning installation of MQAXP V5.0 at 14:45
%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.
******************************************************************************
*            MessageQ for OpenVMS Alpha V5.0                                 *
*                   Installation Procedure                                   *
*            Copyright BEA Systems, Inc. 2000.                               *
*                  All rights reserved.                                      *
*This software is subject to and made available only pursuant to the terms   *
*of the BEA Systems, Inc 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.          *
*                                                                            *
*Use, duplication or disclosure by the U.S. Government is subject to         *
*restrictions set forth in the BEA Systems, Inc 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 Commerical Computer Software -- Licensing clause  *
*at NASA FAR supplement 16-52.227-86; or their equivalent.                   *
*                                                                            *
*MessageQ is a registered trademark of BEA Systems, Inc.                     *
******************************************************************************
```

Step 2—Select kit items for installation

Next specify which parts of the distribution kit you want to install. You can install the Base kit (server, client, and, optionally, the MessageQ/Tuxedo Bridge) or the Client Library kit. You must answer yes to part three to install the BEA MessageQ Client for OpenVMS.

```
=================================================================
The MessageQ for OpenVMS Alpha installation is divided into two parts:
Part 1: Base kit installation
Part 2: Client Library installation
=================================================================
--- Part 1: Select Base Kit Support ---
* Install the Base kit (195000 blks) [NO]: n
* Install the MessageQ/Tuxedo Bridge (1000 blks) [NO]: n
--- Part 2: Select Client Library Kit Support ---
* Install the Client Library kit (12000 blks) [NO]: y
%MQAXP050-I-BLKSREQ, Blocks required for installation is 41000
%MQAXP050-I-DECTCP, DEC TCP/IP support will be built
%MQAXP-I-NODNS, DECdns support will not be installed

* Device where MessageQ will reside [SYS$SYSDEVICE]:
* Do you want to purge files replaced by this installation [YES]?

 ================================================================
 All questions regarding the installation of MessageQ for OpenVMS Alpha
 have now been asked.  The installation will continue for
 approximately 5-60 minutes, depending on load and CPU type.
 ================================================================
.
.
.
```

Step 3—Postinstallation tasks

At the end of the installation procedure, several files are created to store information related to the installation. Once BEA MessageQ software is installed, follow the procedures outlined in the “Postinstallation Tasks” topic in the BEA MessageQ for OpenVMS Installation Guide.

```
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
Installation of MQAXP V5.0 completed at 15:06
    Adding history entry in VMI$ROOT:[SYSUPD]VMSINSTAL.HISTORY
    Creating installation data file: VMI$ROOT:[SYSUPD]MQAXP050.VMI_DATA
VMSINSTAL procedure done at 15:09
```

Recovering from Errors During the Installation

If errors occur during the BEA MessageQ Client installation procedure, recheck your preinstallation steps to ensure that the correct versions of prerequisite software have been installed. Errors can occur during the installation if the following conditions exist:

- The operating system version is not supported by BEA MessageQ Client.
- TCP/IP or DECnet software is not installed or configured (if you are installing over the network).

For descriptions of the error messages generated by these conditions, see the system management documentation for the OpenVMS system that you are using. If an error occurs while installing BEA MessageQ and you believe the error is caused by a problem with the BEA MessageQ Client software, contact technical support using the information provided in the Preface to this manual.

Postinstallation Tasks

After installation is successfully complete, you must:

- run the `DMQ$CL_STARTUP` procedure
- configure the BEA MessageQ Client
- initialize the BEA MessageQ Client environment

Refer to the following topics for more information about these postinstallation tasks.

Run the DMQ$CL_STARTUP Procedure

The command procedure DMQ$CL_STARTUP.COM must be run once after the system is booted to initialize the client run-time environment. The format of this command is:

```
@device:[DMQ$V50.EXE]DMQ$CL_STARTUP
```

where *device* is the device name specified during installation.

DMQ$CL_STARTUP.COM uses the logical name DMQ$CL_TRANSPORT to set up the client environment for use with a specified network transport.

DMQ$CL_TRANSPORT can have one of the following values:

DMQ$CL_TRANSPORT	DMQ$ENTRYRTL points to	RTL Network Support
DECNET	DMQ$DECNET_CLRTLV50.EXE	DECnet only
DEC or UCX (default)	DMQ$UCX_CLRTLV50.EXE	Compaq TCP/IP
PSC	DMQ$PSC_CLRTLV50.EXE	Process TCPware
TGV	DMQ$TGV_CLRTLV50.EXE	PSC Multinet

UCX (Compaq TCP/IP) is used as the default if DMQ$CL_TRANSPORT is undefined.

Configuring the BEA MessageQ Client

Following installation, you must create and configure the BEA MessageQ Client group initialization file. A template file called dmq.ini is located in the following location:

```
DMQ$DISK:[DMQ$V50.USER.CLIENT]DMQ.INI
```

Copy this file to your applications default directory, or a directory shared by multiple users. Configure dmq.ini using the BEA MessageQ Client Configuration utility, dmqclconf. See Chapter 3, “Configuring the BEA MessageQ Client,” for detailed configuration instructions.

If you wish to use a dmq.ini file that is not in your default application directory you can define the logical name DMQCL_INI_FILE to point to the desired dmq.ini file.

To use the BEA MessageQ Client utility programs as documented you must execute the DMQ$CL_SET_LNM_TABLE.COM file as follows:

```
@DMQ$EXE:DMQ$CL_SET_LNM_TABLE
```

This defines DCL symbols for each of the utility programs.

Note: Special care must be exercised when running the BEA MessageQ Server and the BEA MessageQ Client on the same OpenVMS system. It is easy to inadvertantly confuse logical names and their associated tables. The BEA MessageQ Client for OpenVMS user logins must not make reference to the BEA MessageQ Server DMQ$SET_LNM_TABLE or DMQ$STARTUP command procedures. The recommanded method for switching between BEA MessageQ Server and Client applications on an OpenVMS system is for the user to log off, then log back in and execute the DMQ$CL_STARTUP and DMQ$CL_SET_LNM_TABLE to properly configure the logical names.

Initializing the BEA MessageQ Client Environment

The command procedure DMQ$DISK:[DMQ$V50.EXE]DMQ$CL_SET_LNM_TABLE must be executed before developing or running a client application on OpenVMS. The syntax for this command is

```
$ @DMQ$DISK:[DMQ$V50.EXE]DMQ$CL_SET_LNM_TABLE
```

This command procedure defines logical names to identify files or directories used in development and run-time environments. It also defines symbols for the BEA MessageQ Client utility programs. Users can execute this command procedure from their login procedure (LOGIN.COM). This procedure must also be executed for any client application that is run as a detached process or batch job.

3 Configuring the BEA MessageQ Client

This section describes how to configure the BEA MessageQ Client and covers the following topics:

- Configuration Options, Parameters, and Menus
- Configuring the Server Connection
- Configuring Message Recovery Services
- Configuring Logging
- Configuring Tracing
- Testing the Configuration Using the Test Utility

Configuration Options, Parameters, and Menus

Configuring the BEA MessageQ Client is done by editing an initialization file, `dmq.ini`. This file is used at run time by the BEA MessageQ Client. The `dmq.ini` file can be shared by multiple BEA MessageQ-enabled applications using the same general configuration. Individual copies of `dmq.ini` can be used to tailor the configuration for individual applications.

The `dmq.ini` file can be stored in any of the following locations:

- Default working directory where the application is running
- File identified by the DMQCL_INI_FILE logical name

The location of the dmq.ini file determines whether the same configuration is shared by multiple applications. When the application attempts to attach to the BEA MessageQ message queuing bus, the client library searches the directories in the order listed for a copy of the dmq.ini file. The dmq.ini file can be modified using any text editor; however, we recommend using the BEA MessageQ Client Configuration Utility.

Refer to Table 3-1 for the configuration options for the BEA MessageQ Client.

Note: Before using the BEA MessageQ Client utility programs you should execute the following command to define the appropriate symbols:

```
@DMQ$EXE:DMQ$CL_SET_LNM_TABLE
```

Table 3-1 BEA MessageQ Client Configuration Options

Option	Description	Required?
Default Server	Network transport, server host name, and endpoint definition	Yes
Failover	Network transport, server host name, and endpoint definition for the failover server	No
MRS	Settings for enabling the local store-and-forward (SAF) message journal and configuring the local journal files	No
Logging	Settings for logging error events and tracing messages to a log file	No
Tracing	Settings to enable run-time trace information about the API calls and Client library activity	No

To configure the BEA MessageQ Client, use the BEA MessageQ Client Configuration Utility, dmqclconf. The configuration utility is started from the command line using the following command line format:

```
$ dmqclconf [-f file] [-l] [-v] [-h]
```

Refer to Table 3-2 for the command-line parameters.

Table 3-2 BEA MessageQ OpenVMS Command Line Parameters

Option	Description
`-f file`	Specifies the `dmq.ini` file path. The default file is [ ]`dmq.ini`. If this option is not used, `dmqclconf` attempts to locate `dmq.ini` by first looking in the current default directory, then using the `DMQCL_INI_FILE` logical name.
`-l`	Lists the current configuration settings (if used with the `-f file` option), or the default configuration settings
`-v`	Displays the BEA MessageQ Client Configuration Utility version number
`-h`	Displays a brief help message that describes the options for this command

See Listing 3-1 for the BEA MessageQ Client Configuration Utility Main Menu.

Listing 3-1 BEA MessageQ Client Configuration Utility Main Menu

```
Main Menu             (file: []dmq.ini)

  1  Open
  2  Configure
  3  List
  4  Save
  5  Exit

Enter Selection:
```

Refer to Table 3-3 for a description of the BEA MessageQ Client Main Menu Options.

Table 3-3 BEA MessageQ Client Main Menu Options

Option	Description
`Open`	Opens a specific `dmq.ini` file
`Configure`	Configure the BEA MessageQ Client

Table 3-3 BEA MessageQ Client Main Menu Options

Option	Description
`List`	Lists the current (or the default) settings
`Save`	Saves the configuration changes or updates to the `dmq.ini` file
`Exit`	Exits the BEA MessageQ Client Configuration Utility

To begin configuring the BEA MessageQ Client, select item 2, Configure, from the Main Menu. See Listing 3-2 for the BEA MessageQ Client Configure Menu.

Listing 3-2 BEA MessageQ Client Configure Menu Options

```
Configure Menu    (file: []dmq.ini)

  1  Server
  2  Failover
  3  Logging
  4  MRS
  5  Tracing
  6  Previous Menu

Enter Selection:
```

Configuring the Server Connection

Configuring the connection to the BEA MessageQ Client Library Server (CLS) consists of the following two items:

- Default server (required)

  The default CLS is used for all connections to the message queuing bus. Applications that are attempting to connect to a server (or lose a connection to the CLS) attempt to reconnect when the network connection to the server is available. If you do not enable automatic reconnect for the default server, you may want to consider configuring the automatic failover server.

- Automatic failover server

  If the primary default server is not available, the BEA MessageQ Client provides the option of connecting to a failover server to ensure robust client connections. However, if automatic reconnect to the default server is enabled, the automatic failover server cannot be used.

Default Server

The default server identifies the BEA MessageQ server system for all connections to the message queuing bus. If automatic reconnection is enabled, applications that lose a connection to a server (or lose a connection to the CLS) try to reconnect when the network connection to the server is available. Client applications also reconnect in the event that the CLS or host server system is stopped and restarted. During an automatic reconnect event, the BEA MessageQ Client attempts to connect only to the default server. Automatic reconnect does not attempt to use the failover server.

After a successful reconnect, the application is automatically attached to the message queuing bus and messaging operations can continue without interruption. All pending messages in the store-and-forward (SAF) journal are sent to the CLS before new operations can be performed. For example, when a `pams_get_msg` function call triggers the reconnect threshold and a successful automatic reconnect and attach operation completes, the SAF file is completely drained before the `pams_get_msg` function call returns.

The server configuration options shown in Listing 3-3 are described in Table 3-4.

Listing 3-3 Default Server Options

```
Server Configuration
  Network Transport Type (DECnet or TCP/IP) [TCP/IP]:
  Server Hostname [arches]: dmqsrv
  Server Endpoint [5000]:
  Reconnect Interval (# of messages) [0]:
```

Table 3-4 Configuring the Default Server

Option	Description
Network Transport Type	The network-level transport used to send messages to the BEA MessageQ CLS. Choices are either `TCP/IP` or `DECnet`.
Host name	The name of the host running the BEA MessageQ CLS. The name must have a corresponding entry in the local hosts file or DECnet database.
Endpoint	The endpoint used by the BEA MessageQ CLS. The endpoint identifies either the TCP/IP port number or the DECnet object name the CLS uses to listen for client connections. The same endpoint is used when configuring the BEA MessageQ Client to locate the CLS. For DECnet transport, the endpoint can range from 1 to 65535 inclusive. The endpoint value is concatenated with the prefix `DMQCLS_` to create a unique DECnet object name (for example, `DMQCLS_05000`). For TCP/IP transport, the endpoint ranges from 1024 to 65535 inclusive because port numbers less than 1024 are reserved.
Automatic Reconnection	Two alternatives are available to enable automatic reconnection: first, attempt to reconnect after "message interval" operations are made; second, attempt to reconnect every "timer interval" seconds. Only one of the reconnect methods may be used. If both configuration values are non-zero, the message interval is used. If both configuration values are `0`, then automatic reconnection is not enabled.
Use message interval	The number of message operations which occur before attempting to reconnect. If set to `0`, automatic reconnect is not enabled. If the value of the message interval is greater than 0, then automatic failover will not be used. The BEA MessageQ Client attempts to reconnect to the server using the message interval option in the server configuration area as the threshold for making a new connection attempt. Setting the value of the message interval greater than 0 enables the automatic reconnect feature. Any messaging operation call, such as `pams_attach_q`, `pams_put_msg`, or `pams_get_msg`, increments a count used to determine when to attempt another reconnect. When the number of operations attempted reaches the message interval threshold, a reconnect attempt is made. Windows applications can choose to use a higher reconnect value to stream `pams_put_msg` requests into the local journal for forwarding at a later time.

Table 3-4 Configuring the Default Server

Option	Description
Use timer interval	The time period, in seconds, between automatic reconnect attempts. If set to `0`, automatic reconnect is not enabled. If the value of the timer interval is greater than 0, then automatic failover will not be used. The BEA MessageQ Client attempts to reconnect to the server after every timer interval. The reconnect attempts are made independent of the BEA MessageQ calls made by the Client for Windows application. When a connection to the CLS is available, the Client Library automatically issues a `pams_attach_q` operation using the parameters of the first `pams_attach_q` function for a primary queue (PQ) issued by the application.

Automatic Failover Server

If the primary (default) server is not available and automatic failover is enabled, the BEA MessageQ Client provides the option of connecting to a failover CLS. By enabling automatic failover, a BEA MessageQ Client will transparently try to attach to the failover server when the CLS on the primary server group is not available. Attempts to connect to the failover server are only made during a call to `pams_attach_q`. Failover is not used if automatic reconnect to the default server is enabled.

Using the failover capability requires additional planning and work in order for messages to be sent and received correctly. The message queues used by BEA MessageQ Client applications are implemented by the BEA MessageQ server group. The message queues, and any recoverable message journals, are located on the server system.

When connecting to the failover group, the queue address used by the BEA MessageQ Client is likely to change (unless the BEA MessageQ group started on the failover system has the same group ID as the primary server group). Recoverable messages sent to the client using the queue address of the primary server group are not delivered to the client when it reattaches to the failover server in a different BEA MessageQ server group.

The simplest use of automatic failover is when the BEA MessageQ Client attaches to a temporary queue and uses a request/response style of messaging. The client sends requests to one or more servers that send responses back to the queue address that sent

the request. If failover occurs, the BEA MessageQ Client is automatically reattached to a new temporary queue and request messages are sent and responses delivered to the new queue address. The application is unaware that a failover event occurred, except that any pending response is not received.

When clients attach to a specific permanent queue and receive recoverable messages sent to that queue address, they depend on the message queuing resources of that BEA MessageQ group. Recoverable messages sent to the queue address while the client is not attached are saved on that system. If the client reconnects to the same queue name or number, but on a different (failover) BEA MessageQ group, the recoverable messages on the BEA MessageQ group where the default CLS is located are not delivered to the new queue address used by the BEA MessageQ Client.

Automatic failover is not appropriate for all applications. The BEA MessageQ server group and all disk-based queuing resources must also fail over to another system so that messages sent to the BEA MessageQ Client are received after a failover transition. For example, the BEA MessageQ server group can recover by restarting the BEA MessageQ group on another node in an OpenVMS cluster.

See Listing 3-4 for the automatic failover server configuration options.

Listing 3-4 Automatic Failover Server Options

```
Failover Configuration
Enable Automatic Failover (yes/no) [no]: y
  Network Transport Type (DECnet or TCP/IP) [TCP/IP]:
  Server Hostname [oquirh]: dmqbck
  Server Endpoint [5000]:
```

Refer to Table 3-5 for a description of the automatic failover server configuration options.

Table 3-5 Configuring the Automatic Failover Server

Option	Description
Enable Automatic Failover	If set to `yes`, automatic failover is enabled. The failover server is used when the default server is not available and automatic failover is enabled. The use message interval option (see Table 3-4) must be greater than 0.

Table 3-5 Configuring the Automatic Failover Server

Option	Description
Network Transport Type	The network-level transport used to send messages to the BEA MessageQ CLS. Choices are either `TCP/IP` or `DECnet`.
Host name	The name of the host running the BEA MessageQ CLS. The host name must have a corresponding entry in the local hosts file or DECnet database.
Endpoint	The endpoint used by the BEA MessageQ Client and the BEA MessageQ CLS. The endpoint identifies either the DECnet object name or the TCP/IP port number the BEA MessageQ Client used to locate the CLS. The CLS uses the same endpoint to listen for client connections. For DECnet transport, the endpoint can range from 1 to 65535 inclusive. The endpoint value is concatenated with the prefix DMQCLS_ to create a unique DECnet object name (for example, `DMQCLS_05000`). For TCP/IP transport, the endpoint ranges from 1024 to 65535 inclusive. Port numbers less than 1024 are reserved.

Configuring Message Recovery Services

Message Recovery Services (MRS) are the BEA MessageQ services that manage the automatic redelivery of critical messages. Messages that are sent using a recoverable delivery mode are written to the local store-and-forward (SAF) journal when the connection to the server system is not available.

The BEA MessageQ Client ensures delivery of recoverable messages to the CLS on the BEA MessageQ Server by providing a store-and-forward (SAF) journal (`dmqsaf.jrn`) to store recoverable messages when the connection to a CLS is not available. Local SAF journal processing is available when Message Recovery Services (MRS) are enabled in the BEA MessageQ Client configuration. The location of the journal file can be set when configuring MRS.

If MRS is enabled, the message recovery journal is turned on when the client application first initiates an attach operation. If the CLS is not available at the time of an attach, the journal file is opened and the attach operation completes with return a status of `PAMS__JOURNAL_ON`.

When the journal is on, messages sent using the following reliable delivery modes are saved to the journal:

- PDEL_MODE_WF_MEM (using PDEL_UMA_SAF)
- PDEL_MODE_WF_DQF
- PDEL_MODE_AK_DQF
- PDEL_MODE_WF_SAF
- PDEL_MODE_AK_SAF

When the connection to the CLS is reestablished, all messages in the SAF journal are sent before new messages are processed. The SAF messages are transmitted in first-in/first-out (FIFO) order. When the connection to CLS is reestablished, a return status of PAMS__LINK_UP is used to indicate that journal processing is no longer active.

Messages are sent from the SAF when one of the following events occurs:

- The connection to the CLS is established successfully and pending messages exist in the SAF.
- The connection to the CLS is lost and the application continues to send recoverable messages. Additional message operations trigger an automatic reconnect to the CLS that is successful, and messages are pending transmission in the SAF.

The BEA MessageQ Client MRS configuration options allow the SAF journal to be configured as follows:

- A fixed-size file that does not reuse disk blocks
- A fixed-size file that reuses (cycles) disk blocks
- A dynamic file that grows indefinitely until no more disk blocks are available

These options allow you to determine how disk resources are used for message journals. Journal files that grow indefinitely periodically allocate an extent of disk blocks as needed to store messages. When all messages are sent from the SAF and the journal is empty, the disk blocks used by the journal are freed and the journal file returns to its original size.

This portion of the configuration process is *optional* if recoverable messaging is not used. See Listing 3-5 for the MRS configuration options.

Listing 3-5 MRS Configuration Options

```
MRS Configuration
  MRS Enabled (yes/no) [yes]:
  Journal File Directory []:
  Journal File Size (bytes) [48000]:
  Cycle Journal File Blocks (yes/no) [yes]: n
  Fixed Size Journal File (yes/no) [yes]:
  Preallocate Journal File (yes/no) [yes]:
```

Refer to Table 3-6 for the MRS configuration options.

Table 3-6 MRS Configuration Options

Option	Description
MRS Enabled	If set to `yes`, MRS is enabled
Journal File Directory	Specifies the directory where the BEA MessageQ journal file, `dmqsaf.jrn`, is located. The default location is the current working directory.
Journal File Size	Initial size, in bytes, of the journal file
Cycle Journal Blocks	If set to `yes`, the journal *cycles*, (reuses) disk blocks when full and overwrites previous messages. The Cycle Journal Blocks file automatically sets the Fixed Size allocation option. When Cycle Journal Blocks is enabled, all read/write operations to the journal use fixed size journal message blocks.
Fixed Size Journal File	Determines if the journal size is fixed or allowed to grow. If Cycle Journal Blocks is set to `yes`, Fixed Size is also enabled. Journals that do not cycle and are not fixed can grow until the disk is full.
Preallocate Journal	If set to `yes`, the journal file disk blocks are preallocated when the journal is initially opened.
Journal Message Block Size	Defines the file I/O block size, in bytes. Used for journal read/write operations only when Cycle Journal Blocks is enabled. When calculating this value, add 80 bytes to the largest user message (because an 80-byte MRS message header is written to the journal for each user message).

Configuring Logging

The BEA MessageQ Client allows you to log error messages to a file (`dmqerror.log`) It also allows run time message activity to be monitored by writing the messages to a file (`dmqtrace.log`). All log files are located in the current working directory for the application.

Message logging allows you to obtain a complete history of the messaging activity of your application. See Listing 3-6 for the logging configuration options.

Listing 3-6 Logging Options

```
Logging Configuration
  Log Error Events (yes/no) [yes]: y
  Log Messages Sent to Trace File (yes/no) [no]:
  Log Messages Received to Trace File (yes/no) [no]:
```

Refer to Table 3-7 for the message logging and message tracing configuration options. Note that you must perform a `pams_attach_q()` operation for any of the "Log Messages" options to take effect.

Table 3-7 Configuring Message Logging

Option	Description
Log Error Events	If set to `yes`, logs error events to the file `dmqerror.log`. The default behavior is to log error events. If error event logging is enabled, connection errors to the CLS also log the full file path of the configuration file used at the time of the connection attempt. This can help identify problems due to multiple copies of the configuration file.
Log Messages Sent To Trace File	If set to `yes`, sends a copy of BEA MessageQ messages sent by the application to the `dmqtrace.log` message log file.
Log Messages Received To Trace File	If set to `yes`, sends a copy of BEA MessageQ messages received by the application to the `dmqtrace.log` message log file.

Configuring Tracing

Tracing can be a useful debugging tool, because it allows you to observe BEA MessageQ Client processing activity. The trace output may create large output files on your system, and should be used only to monitor specific application behavior. The trace output log file, `dmqcldll.log`, is located in the default working directory for the application. See Listing 3-7 for the tracing configuration options.

Listing 3-7 Tracing Configuration Options

```
Tracing Configuration
  Trace PAMS API Calls (yes/no) [no]: y
  Trace Client Library Activity (yes/no) [yes]: y
```

Refer to Table 3-8 for the tracing configuration options.

Table 3-8 Tracing Configuration Options

Option	Description
Trace PAMS API calls	If set to `yes`, logs the API calls to the file `dmqcldll.log`. The default is no tracing.
Trace Client library activity	If set to `yes`, traces the internal Client activity to the file `dmqcldll.log`. The default is no tracing.

Testing the Configuration Using the Test Utility

To test your newly configured BEA MessageQ Client, run the BEA MessageQ Test Utility `dmqcltest`. The Test Utility is started from the command line using the following command-line format:

```
$ dmqcltest
```

The Test Utility allows you to interactively select the parameter options for individual calls to BEA MessageQ. The program also allows you to test various BEA MessageQ message delivery options and send messages to any process connected to the BEA MessageQ bus. Use the Test Utility for unit testing applications under development.

To use the Test Utility, you first set the parameters associated with an action, then you perform the action. For example, to attach to a queue, you set the desired attach parameters, then execute the attach action.

See Listing 3-8 for the Test Utility main menu options.

Listing 3-8 Test Utility Main Menu

```
Main Menu
 1 Parameters
 2 Actions
 3 Exit
 Enter Menu Selection >> 1
```

Refer to Table 3-9 for the Test Utility Parameters and Actions menu Options.

Table 3-9 Test Utility Parameters and Actions Menu Options

Parameters Menu Options	Actions Menu Options
Attach Parameters	Attach Queue
Bind Parameters	Bind Queue
Cancel Timer Parameters	Cancel Timer
Detach Parameters	Detach Queue
Locate Parameters	Locate Queue
Get Parameters	Get Message
Previous Menu	Previous Menu
Put Parameters	Put Message

Table 3-9 Test Utility Parameters and Actions Menu Options

Parameters Menu Options	Actions Menu Options
Set Timer Parameters	Set Timer
Subscribe MOT Parameters	Subscribe MOT
View Current Parameters	View Current Parameters

The examples in the following figures show how to use the Test Utility to attach to a temporary queue and send a message to another queue. The steps shown by the examples are as follows:

1. Set the Attach parameters to specify a temporary primary queue (Listing 3-9)
2. Set the Put parameters (Listing 3-10)
3. Attach to queue 206 in group 9 (Listing 3-11)
4. Put the message to queue 1 in group 9 (Listing 3-12)
5. Detach from the temporary queue (Listing 3-13)
6. Exit from the Test Utility (Listing 3-14)

Listing 3-9 Specify a Temporary Queue

```
Wed > dmqcltest
Main Menu
 1 Parameters
 2 Actions
 3 Exit
 Enter Menu Selection >> 1
Parameters Menu
 1 Attach Parameters
 2 Bind Parameters
 3 Detach Parameters
 4 Locate Parameters
 5 Put Parameters
 6 Get Parameters
 7 Set Timer Parameters
 8 Cancel Timer Parameters
 9 View Current Parameters
 10 Subscribe MOT Parameters
```

```
 11 Previous Menu
 Enter Menu Selection >> 1
SELECT ATTACH TYPE
       1) Attach Primary
       2) Attach Secondary
Select attach type [1] ?
SELECT ATTACH_MODE
       1) Attach by name
       2) Attach by number
       3) Attach temporary
Select attach mode [3] ?
```

Listing 3-10 Set the Put Parameters

```
Parameters Menu
 1 Attach Parameters
 2 Bind Parameters
 3 Detach Parameters
 4 Locate Parameters
 5 Put Parameters
 6 Get Parameters
 7 Set Timer Parameters
 8 Cancel Timer Parameters
 9 View Current Parameters
 10 Subscribe MOT Parameters
 11 Previous Menu
 Enter Menu Selection >> 4
SELECT PRIORITY
       1) Standard Priority
       2) High Priority
Select priority [1] ?
SELECT DELIVERY MODE
       1) PDEL_MODE_AK_xxx
       2) PDEL_MODE_NN_xxx
       3) PDEL_MODE_WF_xxx

 Select deliver mode [2] ? 3
SELECT DELIVERY MODE
       1) PDEL_MODE_xx_ACK
       2) PDEL_MODE_xx_CONF
       3) PDEL_MODE_xx_DEQ
       4) PDEL_MODE_xx_DQF
       5) PDEL_MODE_xx_MEM
       6) PDEL_MODE_xx_SAF
 Select delivery mode [5] ? 5
SELECT UMA
       1) PDEL_UMA_DISC
```

```
       2) PDEL_UMA_RTS
       3) PDEL_UMA_SAF
       4) PDEL_UMA_DLQ
       5) PDEL_UMA_DLJ
Select UMA [1] ? 1 Enter target group [0] ? 9
Enter target queue [0] ? 1
Enter response queue [0] ?
Enter timeout in seconds [30] ?
Enter message class [1] ? 12
Enter message type [-100] ? 34
Enter message text ? This is a test message from dmqcltest.
```

Listing 3-11 Attach to Queue 206 in Group 9

```
Parameters Menu
 1 Attach Parameters
 2 Bind Parameters
 3 Detach Parameters
 4 Locate Parameters
 5 Put Parameters
 6 Get Parameters
 7 Set Timer Parameters
 8 Cancel Timer Parameters
 9 View Current Parameters
 10 Subscribe MOT Parameters
 11 Previous Menu
 Enter Menu Selection >> 9
Main Menu
 1 Parameters
 2 Actions
 3 Exit
 Enter Menu Selection >> 2
Actions Menu
 1 Attach Queue
 2 Bind Queue
 3 Detach Queue
 4 Locate Queue
 5 Put Message
 6 Get Message
 7 Set Timer
 8 Cancel Timer
 9 View Current Parameters
 10 Subscribe MOT
 11 Previous Menu
```

```
 Enter Menu Selection >> 1
attached to queue 9.206
```

Listing 3-12 Put the Message to Queue 1 in Group 9

```
Actions Menu
 1 Attach Queue
 2 Bind Queue
 3 Detach Queue
 4 Locate Queue
 5 Put Message
 6 Get Message
 7 Set Timer
 8 Cancel Timer
 9 View Current Parameters
 10 Subscribe MOT
 11 Previous Menu
 Enter Menu Selection >> 4
put message to queue 9.1
Actions Menu
 1 Attach Queue
 2 Bind Queue
 3 Detach Queue
 4 Locate Queue
 5 Put Message
 6 Get Message
 7 Set Timer
 8 Cancel Timer
 9 View Current Parameters
 10 Subscribe MOT
 11 Previous Menu
 Enter Menu Selection >> 9
```

Listing 3-13 Detach from the Temporary Queue

```
Main Menu
 1 Parameters
 2 Actions
 3 Exit

 Enter Menu Selection >> 2
Actions Menu
 1 Attach Queue
```

```
 2 Bind Queue
 3 Detach Queue
 4 Locate Queue
 5 Put Message
 6 Get Message
 7 Set Timer
 8 Cancel Timer
 9 View Current Parameters
 10 Subscribe MOT
 11 Previous Menu

 Enter Menu Selection >> 2
detached from queue 9.206
```

Listing 3-14 Exit from the Test Utility

```
Actions Menu
 1 Attach Queue
 2 Bind Queue
 3 Detach Queue
 4 Locate Queue
 5 Put Message
 6 Get Message
 7 Set Timer
 8 Cancel Timer
 9 View Current Parameters
 10 Subscribe MOT
 11 Previous Menu
 Enter Menu Selection >> 9
Main Menu
 1 Parameters
 2 Actions
 3 Exit
 Enter Menu Selection >> 3
```

4 Using the BEA MessageQ Client

This chapter describes how to develop, run, and manage BEA MessageQ Client applications. It contains the following topics:

- Developing Your BEA MessageQ Client Application
- Running Your Application
- Managing Your Application

Note: OpenVMS users must execute: `DMQ$DISK:[DMQ$V50.EXE]DMQ$CL_SET_LNM_TABLE` before building or running client applications. Refer to "Postinstallation Tasks" in Chapter 2 for more information.

Developing Your BEA MessageQ Client Application

This section describes the following special considerations for developing applications to run on the BEA MessageQ Client:

- BEA MessageQ API functions supported by the BEA MessageQ Client
- Limits on API parameter returns imposed by the BEA MessageQ Client
- Contents and location of the BEA MessageQ C/C++ include files

- Considerations for cross-group messaging between systems with different hardware data formats
- Building Client applications
- How to access the sample programs that comes with the BEA MessageQ Client

BEA MessageQ API Support

Table 4-1 shows the API functions supported by the BEA MessageQ Client. A small number of BEA MessageQ API functions are available only for a specific environment and are not supported by the BEA MessageQ Client. For example, the `pams_get_msga` function is available only on OpenVMS systems. To learn about the BEA MessageQ API functions, see the *BEA MessageQ Programmer's Guide*.

Table 4-1 BEA MessageQ Client API Functions

API Function	Description
`pams_attach_q`	Connects a program to the BEA MessageQ bus by attaching it to a message queue
`pams_bind_q`	Binds a temporary queue with a specified queue name
`pams_cancel_select`	Cancels selection of messages using a selection mask
`pams_cancel_timer`	Deletes the specified BEA MessageQ timer
`pams_confirm_msg`	Confirms receipt of a recoverable message
`pams_detach_q`	Detaches a selected message queue, or all attached queues, from the bus
`pams_exit`	Terminates all attachments between the application and the BEA MessageQ bus
`pams_extract_buffer`	Extracts a message from a handle into a message buffer
`pams_get_msg`	Retrieves the next available message from a selected queue
`pams_get_msgw`	Waits until a message arrives in the selected queue, then retrieves the message
`pams_locate_q`	Requests the queue address for a specified queue name

Table 4-1 BEA MessageQ Client API Functions

API Function	Description
pams_put_msg	Sends a message to a target queue
pams_set_select	Defines a message selection mask
pams_set_timer	Creates a timer that sends a message to the application when the timer expires
pams_status_text	Receives the severity level and text description of a user-supplied PAMS API return code
putil_show_pending	Requests the number of pending messages for a list of selected queues

BEA MessageQ Client Function Parameter Limits

The BEA MessageQ Client sets specific limits on function parameter values that allow very large arguments on BEA MessageQ Server systems. The limits for the function parameters reduces the size of network messages exchanged between the BEA MessageQ Client and the remote Client Library Server. Table 4-2 lists the functions, parameters and their maximum values on the BEA MessageQ Client.

Table 4-2 API Function Parameter Maximum Values

API Function	Parameter	Maximum Value
pams_attach_q pams_locate_q	q_name_len	32
pams_attach_q pams_locate_q	name_space_list_len	100
pams_put_msg pams_get_msg	msg_area_size	32,700 (see Note)
pams_detach_q	detach_q options	32
pams_set_select	num_masks	20
putil_show_pending	count	100

Note: Messages larger than 32,700 bytes can be sent or received by using the semantics for large messages (PSYM_MSG_LARGE). Refer to the *BEA MessageQ Programmer's Guide* for information on how to send these kinds of messages.

Include Files for C and C++

The BEA MessageQ Client provides include files for C and C++ language programs. The include files contain the BEA MessageQ API function prototype declarations, return status codes, symbolic constants used for API parameters, and other declarations for using BEA MessageQ message-based services. Table 4-3 lists the standard BEA MessageQ include files, which are described in the *BEA MessageQ Programmer's Guide*. The default location for these include files is:

```
DMQ$DISK:[DMQ$V50.USER.CLIENT]
```

Table 4-3 C Language Include Files

Include File	Contents
p_entry.h	Function prototypes and type declarations for the BEA MessageQ API
p_group.h	Constant definitions for BEA MessageQ message-based services
p_msg.h	Constant definitions for message-based services
p_proces.h	Constant definitions for BEA MessageQ (for OpenVMS) processes
p_return.h	Return status values
p_symbol.h	Symbolic constants used for function parameters
p_typecl.h	Constant definitions of BEA MessageQ message type and class for message-based services
fml32.h	Constant definitions and type declarations for support of FML32 (Field Manipulation Language).
tmenv.h	Constant definitions for support of FML32 (Field Manipulation Language).

BEA MessageQ Client Return Codes

All BEA MessageQ return codes are defined in the include file, `p_return.h`. Some of the return codes are specific to the BEA MessageQ Client and are not returned to server-based applications.

Table 4-4 lists the return codes specific to the BEA MessageQ Client.

Table 4-4 BEA MessageQ Client Return Codes

Return Code	Description
PAMS__JOURNAL_FAIL	The MRS service could not add messages to the local journal because of an operating system I/O error.
PAMS__JOURNAL_FULL	The MRS service could not add messages to the local journal because it is full.
PAMS__JOURNAL_ON	The link to the CLS is broken and the MRS service reports that journaling has begun.
PAMS__LINK_UP	The link to the CLS has been reestablished.
PAMS__NETERROR	The network connection to the CLS is broken.
PAMS__NETNOLINK	The network connection to the CLS is not available.
PAMS__PREVCALLBUSY	A previous BEA MessageQ function call is still in progress.

There are platform-specific differences in the numeric values for the `p_return.h` return codes. The OpenVMS version of `p_return.h` contains numeric values different from those used on Windows NT or UNIX. Client applications do not need to be concerned with these differences because the BEA MessageQ Client returns status codes as they are defined on the client system, regardless of the system where the CLS is running.

It is recommended that programs use the symbolic value when testing the return status codes, rather than a numeric value. For example,

```
if ( status == PAMS__NETNOLINK )
```

instead of

```
if ( status == -278 )
```

This improves code portability because of the platform-specific differences in the numeric values listed in `p_return.h`. It also makes code maintenance easier in the event that any status code numeric value is changed.

Byte Order Considerations for Application Developers

BEA MessageQ provides the capability to send and receive messages between many different types of operating systems and CPU architectures. The byte order used by different CPU architectures is referred to as either little endian (or right-to-left order) or big endian (left-to-right order). Application designers must take into account the differences in byte ordering when designing a distributed application with BEA MessageQ.

The byte order used on the BEA MessageQ Client system and the CLS platform may be different. For example, a Windows PC with an Intel x86 CPU is a little endian machine and an HP PA-RISC system is a big endian machine. This means that integer values sent in the message area from the client are represented differently when received by the application server on the host.

The BEA MessageQ Client and CLS handle differences in byte ordering by using network byte order when the Client and Server system are based on different representations (network byte order is big endian.). This ensures that the arguments to the BEA MessageQ API functions called on the client are passed correctly to CLS platform to initiate the messaging operation.

Note: The BEA MessageQ Client **does not** perform byte-swapping on the user data passed in the static buffer message area for `pams_put_msg` or `pams_get_msg` calls. Only BEA MessageQ self-describing messages, which use FML buffers, perform data marshaling between systems with unlike endian formats. Refer to the *BEA MessageQ Programmer's Guide* for more information about how to use self-describing messages.

There are various techniques for handling the byte order differences in the client or server application components:

- One approach is to send user data messages containing only character string data. Integer values are converted to the corresponding character representation before they are sent in the message.

- Another approach is to design an application-specific interface for sending and receiving messages that implements marshaling routines for the user data contained in each message.

The data marshaling routines can be implemented as a set of library routines designed specifically to support data format conversion. These routines are typically written so that each marshal routine performs one specific record conversion. Standard socket routines are available to support byte-order conversion. These routines are `htonl`, `htons`, `ntohl`, and `ntohs`. For example, `htonl` means host to network long (32-bit) conversion.

Building Your Client Applications

On OpenVMS systems, developers use the following syntax to compile BEA MessageQ Client C application programs:

```
$ cc MY_DMQ_PROGRAM/INCLUDE=DMQ$USER:
```

The `/INCLUDE` option for this command is used to specify the location of BEA MessageQ header files.

Client applications can be linked with either:

- the BEA MessageQ Client run-time library (RTL), or
- the BEA MessageQ object library (OLB).

Linking applications with the BEA MessageQ RTL allows code sharing between numerous simultaneous users. It also saves memory, disk space, and link time. Applications linked with the BEA MessageQ for OpenVMS RTL can run in the BEA MessageQ for OpenVMS Server environment simply by redefining the `DMQ$ENTRYRTL` logical name.

Applications linked with the BEA MessageQ OLB must be relinked if they are moved to a BEA MessageQ Server system. The files required for linking are located in the `DMQ$LIB` and `DMQ$USER` directories.

Linking Applications with the BEA MessageQ Runtime Library

To link an application with the RTL, use the DMQ$LIB:DMQ/OPT switch in the linker command line. For example, to link a program called MY_DMQ_PROGRAM with the BEA MessageQ Client RTL you would use the following command:

```
$ link MY_DMQ_PROGRAM, DMQ$LIB:DMQ/OPT
```

The options file contains all the commands needed to build an application with the BEA MessageQ Client RTL. The DMQ$LIB:DMQ.OPT file uses the DMQ$ENTRYRTL logical to identify the RTL.

The choice of RTL will determine which network transports are available for the client application. For example, DMQ$EXE:DMQ$DECNET_CLRTLV50.EXE supports only DECnet.

If you have TCP/IP installed, the installation procedure will link an RTL supporting each TCP/IP stack and DECnet. The RTL names and supported transports are shown in the following table.

Table 4-5 BEA MessageQ Client for OpenVMS RTL Names

RTL Name	Transport Supported
DMQ$DECNET_CLRTLV50.EXE	DECnet only
DMQ$UCX_CLRTLV50.EXE	Compaq TCP/IP
DMQ$TGV_CLRTLV50.EXE	Process Software Corporation (PSC) Multinet TCP/IP
DMQ$PSC_CLRTLV50.EXE	Process Software Corporation (PSC) TCPWare TCP/IP

Linking Applications with the BEA MessageQ Object Library

The syntax for linking an application with the BEA MessageQ Client OLB is shown in the following example:

```
$ link MY_DMQ_PROGRAM, DMQ$LIB:CL.OLB/LIBRARY/INCLUDE=(QIO TCPIP option)
```

In the previous command, the /INCLUDE=(QIO_TCPIP_*option*) is used to select the desired TCP/IP stack. The following table shows the possible choices for QIO_TCPIP_*option*.

Table 4-6 BEA MessageQ Client for OpenVMS TCP/IP Option Names

TCP/IP Option	Transport Supported
/INCLUDE=(QIO_TCPIP_STUB)	DECnet only
/INCLUDE=(QIO_TCPIP_UCX)	Compaq TCP/IP
/INCLUDE=(QIO_TCPIP_TGV)	Process Software Corporation (PSC) Multinet TCP/IP
/INCLUDE=(QIO_TCPIP_PSC)	Process Software Corporation (PSC) TCPWare TCP/IP

Sample Programs

The BEA MessageQ Client is distributed with a number of sample application programs that demonstrate many features of the BEA MessageQ API. If the sample programs were selected during installation, they will be located in the BEA MessageQ installation directory tree. The default location is:

```
DMQ$EXAMPLES
```

The sample programs consist of C language source modules. The sample programs are identical to the sample programs distributed with the BEA MessageQ Server products, which demonstrate the portability of the BEA MessageQ API across all supported platforms.

The command procedure DMQ$EXAMPLES:X_BUILD.COM is used to build the sample programs. Copy the X_*.C sample programs and X_BUILD.COM from DMQ$EXAMPLES to a personal development directory before modifying any of the files. Use the following command to compile and link the sample programs.

```
$ @x_build
```

Note that the command procedure DMQ$DISK:[DMQ$V50.EXE]DMQ$CL_SET_LNM_TABLE must be executed before developing or running a client application on OpenVMS. The syntax for this command is:

```
$ @DMQ$DISK:[DMQ$V50.EXE]DMQ$CL_SET_LNM_TABLE
```

This command procedure defines logical names to identify files or directories used in development and run-time environments. It also defines symbols for the BEA MessageQ Client utility programs. Users can execute this command procedure from their login procedure (`LOGIN.COM`). This procedure must also be executed for any client application that is run as a detached process or batch job.

Client applications are linked with the BEA MessageQ Client run-time library (RTL). Run-time libraries allow code sharing between numerous simultaneous users. It also saves memory, disk space, and link time. The files required for linking are located in the `DMQ$LIB` and `DMQ$USER` directories.

To link an application, use the `DMQ$LIB:DMQ/OPT` switch in your linker command line. For example, to link a program called `MY_DMQ_PROGRAM` with the BEA MessageQ Client, you would use the following command:

```
$ link MY_DMQ_PROGRAM, DMQ$LIB:DMQ/OPT
```

The options file contains all the commands needed to build an application with the BEA MessageQ Client RTL.

Running Your Application

This topic explains how to run your application with BEA MessageQ Client. Before attempting to run a BEA MessageQ Client application, verify that the TCP/IP or DECnet connection between the BEA MessageQ Client and Server is properly configured. Use the `ping` utility to check the TCP/IP connection (see the documentation for TCP/IP networking for your system for more information). Use the `ncp tell` command to check the DECnet connection.

To use the BEA MessageQ Client, your run-time environment must meet the following software requirements:

1. The BEA MessageQ for UNIX, Windows NT, or OpenVMS product must be installed on a server system. A message queuing group must be configured to support the requirements of your messaging application environment. The BEA MessageQ Client applications use messaging resources, including message queues,

message buffers, and system resources, on the server system. See the *Installation and Configuration Guide* for your BEA MessageQ Server system and review the system resource requirements for using BEA MessageQ in your environment.

2. If you are planning to use the TCP/IP transport, the host names for the client and server systems must be properly identified in the `hosts` files on both the BEA MessageQ Client and Server. Table 4-7 shows the location of host files on all BEA MessageQ Servers.

Table 4-7 Hosts File Location

Systems	Hosts File Location
UNIX	`/etc/hosts`
Windows 95	See your TCP/IP vendor documentation.
Windows NT	`c:\winnt\system32\drivers\etc\hosts`
OpenVMS	Use the TCP/IP for OpenVMS configuration utilities. Refer to your vendor documentation.

3. If you are planning to use the DECnet transport for communication between the BEA MessageQ Client and the CLS, the DECnet node names must be defined on the client and server systems. The `ncp tell` command can be used to verify that the client and server systems are connected. Refer to the DECnet documentation for more information about how to configure DECnet networks.

For a complete description of BEA MessageQ Server and TCP/IP transports supported by the BEA MessageQ Client, see the *BEA MessageQ Release Notes*.

Run-time Files

The run-time configuration file, `dmq.ini`, is required to run a client application. This file can be located in the application's working directory, or in a location identified by the `DMQCL_INI_FILE` logical name. The definition of `DMQCL_INI_FILE` must include the file name. For example, if several users want to share a `dmq.ini` file located in a directory called `TEST$DISK:[TEST_ENV.DMQ]`, they would each define `DMQCL_INI_FILE` as follows:

```
$ define   DMQCL_INI_FILE    TEST$DISK:[TEST_ENV.DMQ]DMQ.INI
```

Managing Your Application

The BEA MessageQ Client includes several utility programs for testing client applications and managing the BEA MessageQ Client environment. The default location for the utilities is:

```
DMQ$EXE
```

Refer to Table 4-8 for a list of the BEA MessageQ Client utilities.

Table 4-8 BEA MessageQ Client for OpenVMS Utility Programs

Utility Program	Filename	Description
Test utility	`dmq$cltest.exe`	An interactive application for sending and receiving messages. Refer to Chapter 3, "Configuring the BEA MessageQ Client" for a description of how to run the Test utility.
MRS utility	`dmq$clmrsu.exe`	Displays the contents of the local store-and-forward (SAF) journal. Refer to "MRS Utility" for information on how to run the MRS utility.
Configuration Utility	`dmq$clconf.exe`	Allows configuration of the BEA MessageQ Client. Refer to Chapter 3, "Configuring the BEA MessageQ Client" for detailed configuration information.

MRS Utility

The BEA MessageQ Client MRS utility lets you view the contents of local SAF journals. When a sender program running on the BEA MessageQ Client sends a message marked as recoverable, it is written to the SAF journal on the client system. In the event that the recoverable message cannot be delivered to the CLS on the BEA MessageQ Server and stored by the BEA MessageQ message recovery system, it can be resent at a later time from the SAF journal on the BEA MessageQ Client using this utility.

The MRS utility is started with the following command:

```
$ dmqclmrsu [-h | -v | -d | -l | -n message | -t message]
  [-f saf_path]
```

Table 4-9 MRS Utility Command Line Parameters

Command Switch	Description
`-h`	Displays a brief help message
`-v`	Display MRS utility version number
`-d`	Display journal file header details
`-l`	Brief display of all messages in the journal
`-n` *`message`*	Display detail for the specified message
`-t` *`message`*	Transmit the specified message
`-f` *`saf_path`*	Specifies the full file path to the desired journal file. The default is `./dmqsaf.jrn`

Listing 4-1 shows the BEA MessageQ Client MRS utility using the `-l` and the `-n` options.

Listing 4-1 BEA MessageQ Client MRS Utility

```
$dmqclmrsu -l

SAF journal: dmqsaf.jrn
 Msg     Source       Target      Class   Type   Pri   Size     Data...
-----  -----------  -----------  ------  ------  ---  ------  ------------
    1      0.0          5.300        66      99   0       13  'first mess'
    2      0.0          5.300        66      99   0       14  'second mes'
    3      0.0          5.300        66      99   0       13  'third mess'
    4      0.0          5.300        66      99   0       14  'fourth mes'
    5      0.0          5.300        66      99   0       13  'fifth mess'
    6      0.0          5.300        66      99   0       13  'sixth mess'
    7      0.0          5.300        66      99   0       15  'seventh me'
    8      0.0          5.300        66      99   0       14  'eighth mes'
    9      0.0          5.300        66      99   0       13  'ninth mess'
   10      0.0          5.300        66      99   0       13  'tenth mess'

$dmqclmrsu -n 7

SAF journal: dmqsaf.jrn
  Detail of message: 7

  Source:      0.0       Priority: 0       Size: 15      Large_Size: 0
  Target:      5.300     Class:    66      Type: 99
  Resp Q:      0.0       Delivery: 29      UMA:  5       Timeout: 100

  Contents of message buffer:
   XB 73, 65, 76, 65, 6E, 74, 68, 20, 6D, 65   ! 'seventh me'
   XB 73, 73, 61, 67, 65                       ! 'ssage'
```

5 Troubleshooting

This chapter describes how to identify and correct problems while running your BEA MessageQ client applications. Troubleshooting includes the following topics:

- Determining the Version Number of the Client
- Identifying Run-Time Errors
- Logging an Error Event
- Failing to Connect to the CLS
- Identifying Network Errors
- Decoding TCP/IP Error Codes
- Tracing PAMS API Activity
- Tracing Client Library Activity
- Recovering from Client Crashes

Determining the Version Number of the Client

To obtain technical support, you must know the version number of the BEA MessageQ Client software the you are running. To determine the version of the BEA MessageQ Client library, enable tracing of Client Library activity, run your application, and check the trace file `dmqcldll.log` for the version number.

Identifying Run-Time Errors

Problems at runtime can arise from a variety of error conditions. To identify and solve problems with the BEA MessageQ Client, you can use a variety of tools to track down the source of the problem. The following list provides some ideas to help you to help you troubleshoot the source of application problems:

- Check the contents of the `dmqerror.log` file to get more information about the problem. Network errors are identified in the error log file.
- Use the trace output capability on the BEA MessageQ Client and the Client Library Server (CLS) to get a detailed flow of the activity that leads up to the problem. For shorter log files, use the `PAMS_TRACE` environment variable on the BEA MessageQ Client.
- Use the TELNET Utility to log in to the remote system and run the BEA MessageQ Monitor Utility. On UNIX systems, use the character-cell program, `dmqmonc`, to monitor BEA MessageQ groups remotely.
- Use the network utilities provided by the stack vendor to monitor the network connections between the client and the server. For example, on UNIX, use `netstat` on the server system to monitor the TCP/IP connections on the host system where the CLS is running.
- Try to repeat the error using the Test Utility included with the BEA MessageQ Client. Reproducing problems with the Test Utility is an effective way to isolate application programming errors and provide a convenient way to test problems.

This chapter summarizes how to find and resolve application problems.

Logging an Error Event

Run time errors detected by the Client library are written to the `dmqerror.log` file in the default directory for the application. The errors indicate a run-time problem due to either a configuration error, an application error, network problem, or unexpected server response.

Error event logging can be either enabled or disabled by changing the BEA MessageQ Client Configuration Logging option. When error event logging is disabled, the `dmqerror.log` file is not used and no information on error conditions is available. Refer to Configuring Tracing in Chapter 3 for more information about trace file settings.

Failing to Connect to the CLS

The BEA MessageQ Client for UNIX attempts to establish a connection to the CLS in response to a call to `pams_attach_q`.

When the connection attempt fails, `pams_attach_q` returns the following error status:

```
PAMS__NETNOLINK -278
```

Check the file `dmqerror.log` for the full path of the configuration file (`dmq.ini`) used, the host name, and the endpoint of the server system with which the BEA MessageQ Client attempted to connect. Refer to "Identifying Run-Time Errors" for additional information about the `PAMS_NETNOLINK` error.

Identifying Network Errors

Network errors result from the Client Library receiving an error when attempting to read or write on the network link. Occasional network connection problems can occur due to the state of the TCP/IP protocol stack or the network connection to the host system. Network errors are identified by the return status from the `pams_attach_q` function, such as the following:

```
PAMS__NETNOLINK -278
```

Network connection errors might also occur when attempting to execute any of the BEA MessageQ API functions. For example, the `pams_put_msg` and `pams_get_msg` functions return the following return code when the connection to the server is broken and MRS is not enabled:

```
PAMS__NETERROR -276
```

The specific steps for clearing the network error depend on how the problem developed. The following actions will generally clear the problem:

1. Check the error event log file, `dmqerror.log`, for a description of the error event.
2. Stop and restart the application. In some cases, restarting the application or simply retrying the attach operation succeeds.
3. Stop and restart the CLS.

Decoding TCP/IP Error Codes

Refer to Table 5-1 for the common TCP/IP error status codes logged to the file `dmqerror.log` when using the TCP/IP transport.

Table 5-1 Common TCP/IP Connect Errors

TCP/IP Error	Meaning	Recovery
`EINVAL`	A socket initialization error	Verify that the TCP/IP stack was installed correctly.
`ECONNREFUSED`	Connection request to the server is refused because a CLS is not listening on the endpoint	Verify that the endpoint defined in the Client for UNIX Server configuration endpoint matches the endpoint used by the CLS. Verify that the CLS is running.
`ENOBUFS`	The TCP/IP stack has run out of available network buffers.	Frequent attach/detach operations may result in connections that have not cleared out. Usually, they clear after a 60-90 second time-out. Increase the maximum number of TCP/IP sockets if needed.

Tracing PAMS API Activity

To obtain a time-stamped output file showing the sequence of BEA MessageQ function calls and return status codes, follow these steps:

1. Invoke the Configuration Utility.
2. Choose Configure from the Main menu, then choose Tracing from the Configure menu.
3. Set the Trace PAMS API Calls option to `yes`.

The information from the `pams_` function call trace is written to the `dmqcldll.log` file in the default directory for the application. The PAMS tracing option can be used to observe the sequence of message function calls to determine the run-time behavior of the application.

Tracing Client Library Activity

To obtain detailed, time-stamped traces of the Client Library activity, follow these steps:

1. Invoke the Configuration Utility.
2. Choose Configure from the Main menu, then choose Tracing from the Configure menu.
3. Set the Trace Client Library Activity option to yes.

The information from the library trace might be useful to debug connection problems between client library applications and the CLS. The library trace output is written to the log file, `dmqcldll.log`, in the default directory for the application. Be aware that the output from the tracing option can become very large over a long period of time.

A CLS server trace might be useful to get a detailed time-stamped trace of the client requests and BEA MessageQ message operations performed by the CLS. For more information about trace output from the CLS, refer to the installation and configuration information for your BEA MessageQ server platform.

Recovering from Client Crashes

Occasionally, applications crash (particularly during development) and do not have an opportunity to close or return resources in use before terminating. Applications using the BEA MessageQ Client that are attached to the message queuing bus and then crash (or terminate) without calling `pams_exit` or `pams_detach_q`, leave many resources allocated but not available for reuse.

Resources that are in use after a client application crash include:

- Global memory allocated on behalf of the client application
- Network protocol resources, such as sockets
- Network resources on the server system
- Message queue resources used by the CLS on behalf of the client

After the client crashes, the server system still has an open connection to the client and the CLS remains attached to the primary queue used by the client. The network protocol keep-alive mechanism does not notify the server that the client has gone away for a lengthy time period. Typically, you can reboot the client system and the server still functions as though it has a connection open to the client.

Restarting the client application usually establishes a new connection to CLS. If network connect errors occur, follow the troubleshooting procedure described in the "Identifying Network Errors" topic. The procedure releases and frees all resources used by the client.

If the client application calls `pams_attach_q` using either `ATTACH_BY_NAME` or `ATTACH_BY_NUMBER` to attach to a specific primary queue, the CLS detects a client reconnect attempt and automatically terminates the CLS instance (server process or thread) attached to the same message queue. Reconnecting to the same queue is only accepted if the client application is attempting to reconnect from the same host as the previous connection.

If the client application calls `pams_attach_q` using the `ATTACH_TEMPORARY` attach mode, a new instance of the CLS is started to support the client reconnect. The previous instances of the CLS remains active. For information about terminating CLS servers, see the CLS topic in the installation and configuration information for your BEA MessageQ server platform.

Index