BEA MessageQ

MVS Client
User's Guide

BEA MessageQ MVS Client Version 5.0
Document Edition 5.0
February 1999

Copyright

Restricted Rights Legend

Trademarks or Service Marks

BEA MessageQ MVS Client User's Guide

Document Edition	Date	Software Version
5.0	February 1999	BEA MessageQ MVS Client, Version 5.0

Contents

Preface

1. Introduction

2. Configuring the BEA MessageQ MVS Client

3. Building Your Application

4. Running Your Application

5. Troubleshooting

Index

Preface

Purpose of This Document

This document describes the BEA MessageQ MVS Client product and provides the information you need to configure and develop BEA MessageQ applications that run in an IBM MVS environment. BEA MessageQ MVS applications fully support BEA MessageQ features using the BEA MessageQ Client Library Server software running on a BEA MessageQ Server system.

Who Should Read This Document

This document is intended for system administrators, network administrators, and application developers who are interested in integrating their MVS mainframe environments with distributed heterogeneous systems through the use of BEA MessageQ middleware.

How This Document Is Organized

The *BEA MessageQ MVS Client User's Guide* is organized as follows:

- Chapter 1, "Introduction," describes the MVS Client product and its benefits. This chapter also provides an overview of the BEA MessageQ MVS Client architecture.

- Chapter 2, “Configuring the BEA MessageQ MVS Client,” provides instructions for creating the MVS Client Datasets, configuring MVS Client software, and testing the configuration.
- Chapter 3, “Building Your Application,” describes how to build an application to use the BEA MessageQ MVS Client library.
- Chapter 4, “Running Your Application,” provides information for running your application in MVS batch and CICS environments.
- Chapter 5, “Troubleshooting,” describes how to identify and correct problems while running your application.

How to Use This Document

This document, *BEA MessageQ MVS Client User’s Guide*, 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

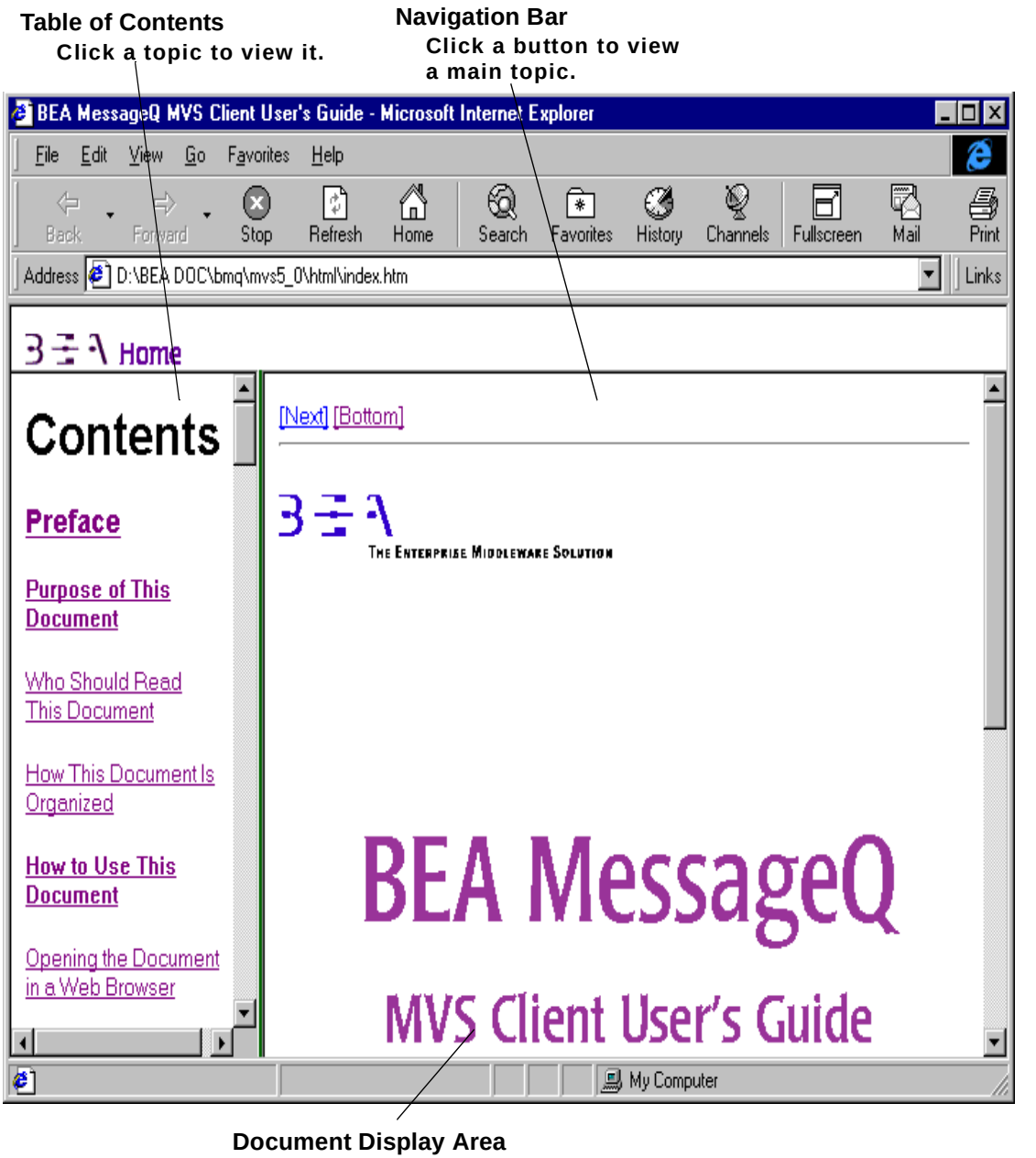
To access the online version of this document, open the following HTML file in a Web browser:

```
/doc/bmq/mvs5_0/usergde/index.htm
```

Note: The online documentation requires a Web browser that supports HTML version 3.0. Netscape Navigator version 2.02 or Microsoft Internet Explorer version 3.0 or later are recommended.

Figure 1 shows the online document with the clickable navigation bar and table of contents.

Figure 1 Online Document Displayed in a Netscape Web Browser



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 IBM Connectivity Products CD and the BEA MessageQ Online Documentation CD also include 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 *Introduction to Message Queuing.*
Ctrl+Tab	Indicates that you must press two or more keys sequentially.
italics	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.
	Examples:
	`INCLUDE SYSLIB`
	`pams_attach_q`
	`\bmq\mvs5_0\include`
	`.htm`
	`bmq.doc`
	`BITMAP`
	`float`

Convention	Item
`monospace boldface text`	Identifies significant words in code. *Example*: `put_msg (msg_ptr, class, type)`
`monospace italic text`	Identifies variables in code. *Example*: `String expr`
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: 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. *Example*: `int32 pams_get_msg (msg_area, priority ... [sel_filter] [psb] [show_buffer] ...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	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*: `int32 pams_get_msg (msg_area, priority ... [sel_filter] [psb] [show_buffer] ...`
. . .	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 MVS Client software.

BEA MessageQ MVS Client Documentation

The BEA MessageQ MVS Client information set consists of the following documents:

BEA MessageQ MVS Client Installation Guide

BEA MessageQ MVS Client User's Guide

BEA MessageQ MQSeries Connection and MVS Client Release Notes

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.

BEA Publications

You may find the following BEA MessageQ publications helpful when using BEA MessageQ MVS Client:

BEA MessageQ Introduction to Message Queuing

BEA MessageQ Programmer's Guide

BEA MessageQ Installation and Configuration for Windows NT

BEA MessageQ Installation and Configuration for UNIX

BEA MessageQ System Messages

BEA MessageQ FML Programmer's Guide

BEA MessageQ Reference Manual

BEA MessageQ Client for Windows User's Guide

BEA MessageQ Client for UNIX User's Guide

Other Publications

You may also find the MVS documentation helpful. For information on MVS, see the IBM MVS documentation set.

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 MVS Client, or if you have problems installing and running BEA MessageQ MVS Client, 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 chapter provides an overview of the BEA MessageQ MVS Client software. It includes the following sections:

- What Is the MVS Client?
- Benefits of Using the MVS Client
- Architectural Overview

What Is the MVS Client?

The BEA MessageQ MVS Client, Version 5.0, is a client implementation of the BEA MessageQ Application Programming Interface (API). The MVS Client relies on a BEA MessageQ server group to provide complete message queuing. The MVS Client uses an internal request/response protocol to communicate with a Client Library Server (CLS) running on a BEA MessageQ server system. The MVS Client is connected to the message queuing bus by a network connection between the BEA MessageQ MVS Client library and the CLS process in the remote BEA MessageQ server group.

The CLS is a BEA MessageQ application that runs as a background server to handle multiple MVS Client connections. The CLS performs all communication with the MVS Client application until the application detaches from the message queuing bus. The message queue operations and network activities of each client are isolated from other clients.

The CLS acts as a remote agent to perform BEA MessageQ message functions in the BEA MessageQ server group on behalf of the MVS Client. The MVS Client library establishes a network connection to the CLS when an application initiates an attach queue operation. The connection to the CLS is closed when the application performs a detach queue operation.

The BEA MessageQ MVS Client library allows multiple applications to connect to separate queues on the message queuing bus. This library creates a separate network connection to the CLS for each application using BEA MessageQ. The total number of applications that can connect to the message queuing bus is limited by the number of TCP/IP sessions.

The MVS Client provides a local store-and-forward (SAF) journal to store recoverable messages when the connection to the CLS is not available. When the connection to the CLS is reestablished, all messages in the SAF journal are sent before new messages are processed.

Benefits of Using the MVS Client

The BEA MessageQ MVS Client provides application developers a means to integrate their MVS mainframe environments with distributed heterogeneous systems through the use of BEA MessageQ message queuing middleware. The BEA MessageQ MVS Client, with support for MVS CICS and MVS batch, provides guaranteed message delivery between IBM mainframe applications and others running on any BEA MessageQ supported platforms by means of a direct TCP/IP network connection.

The BEA MessageQ MVS Client allows users to make BEA MessageQ calls from MVS mainframe applications, providing customers the ability to use a single API for all application environments. The wide range of platforms covered by BEA MessageQ offers developers a single, consistent programming interface to develop and deploy distributed applications across the entire spectrum of enterprise systems, including Microsoft Windows 95 desktops, Windows NT, all popular UNIX variants, OpenVMS, and IBM MVS mainframe environments.

The BEA MessageQ MVS Client can host its communications capabilities on a BEA MessageQ server implementation residing on any node in the network. The value of off-loading the BEA MessageQ communications capability to a non-IBM MVS node, while making BEA MessageQ calls and offering a recoverable link from the MVS

environment, is important for companies looking to integrate the applications running on their MVS mainframes with applications running on UNIX, OpenVMS, and Windows NT servers.

Architectural Overview

The BEA MessageQ MVS Client, Version 5.0, is a client implementation of the BEA MessageQ Application Programming Interface (API). It relies on a BEA MessageQ server group to provide complete message queuing. If the network connection to the BEA MessageQ server group is lost or unavailable, the BEA MessageQ MVS Client optionally stores messages in a local journal file for later retransmission.

The BEA MessageQ MVS Client provides an object library (`DMQ.V5R0.LIB`) to support BEA MessageQ enabled applications. In addition, the media includes JCL procedures (`DMQ.V5R0.CNTL`) that are examples to be used in configuring and building your BEA MessageQ applications with the MVS Client. For a list of the JCL procedures, see "Creating MVS Client Datasets" in Chapter 2.

What Is the Client Library Server?

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

On server systems where both the operating system and the BEA MessageQ implementation support multithreading (such as BEA MessageQ on Windows NT), the CLS is multithreaded. On UNIX server systems, the CLS uses a separate process to handle each MVS Client connection. On OpenVMS server systems, the CLS can operate in the following 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.

When the server starts, it initializes a listener process (or thread) that establishes a network endpoint and waits for connections from MVS Clients. The endpoint on which the CLS listens is determined by the command-line arguments used to start the server. MVS Clients attempt to connect to the CLS when they initiate an attach queue operation.

The MVS Client library uses configuration information in the BEA MessageQ Server group configuration file to identify the location of the CLS. The CLS creates a server subprocess (or server thread with a client context variable) for each new client connection. The server process or thread terminates when the client detaches from the bus, or the network connection is closed.

How Do the MVS Client and CLS Work Together?

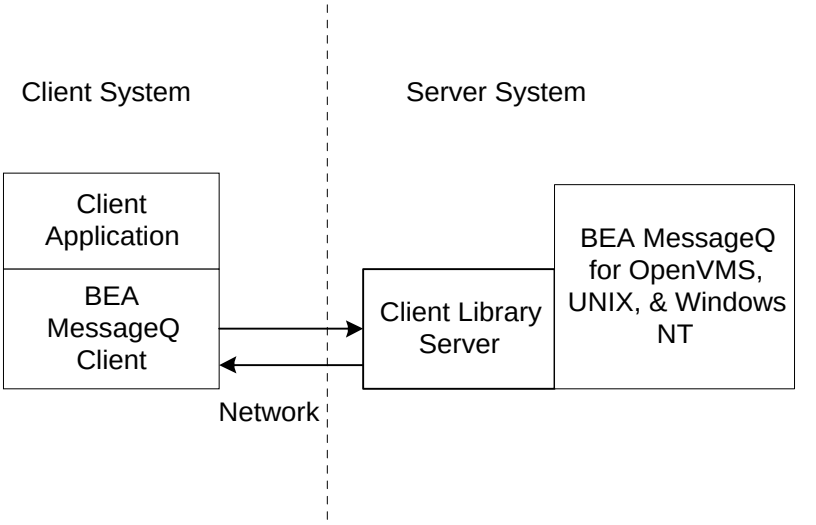
The BEA MessageQ MVS Client uses an internal request/response protocol to communicate with the CLS running on a BEA MessageQ Server system. The BEA MessageQ MVS Client provides a “lightweight” client connection to the BEA MessageQ message queuing bus.

The implementation is considered lightweight because:

- The message queues for all MVS Client applications are implemented on a remote system running a BEA MessageQ server group.
- Message delivery to the destination processes is provided by the message queuing engine, and is implemented by the full-function BEA MessageQ Server group.
- Message routing and cross-group transport among multiple BEA MessageQ server systems and other MVS Clients is provided by the BEA MessageQ Server group.
- Full message recovery services for guaranteed message delivery are provided by the MRS capability of the BEA MessageQ Server group. The BEA MessageQ MVS Client provides a local SAF journal for sending messages when the connection to the BEA MessageQ Server group is not available.

The MVS Client connection to the message queuing bus is provided by a network connection between the BEA MessageQ MVS Client library and a CLS process in the remote BEA MessageQ server group. Figure 1-1 shows the relationship of the MVS Client and CLS to the BEA MessageQ message queuing bus.

Figure 1-1 BEA MessageQ Client and Server Components



The CLS acts as a remote agent to perform BEA MessageQ messaging functions in the BEA MessageQ Server group on behalf of the MVS Client. The MVS Client library establishes a network connection to the CLS when an application initiates an attach queue operation. The connection to the CLS is closed when the application performs a detach queue operation.

How the MVS Client Uses BEA MessageQ Function Calls

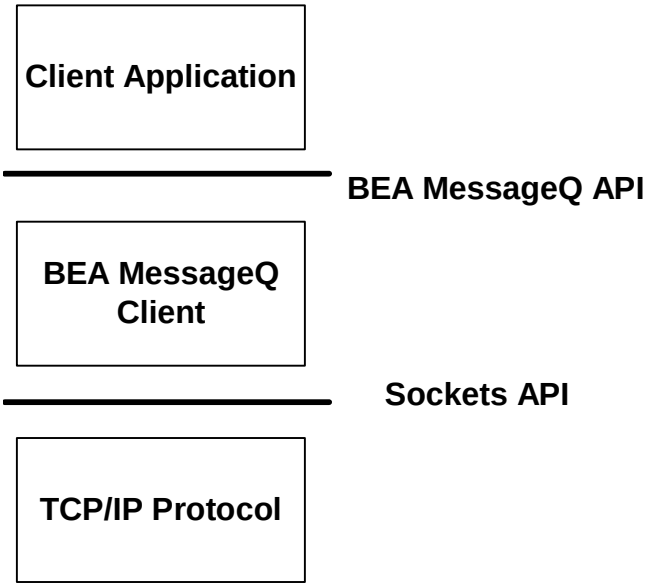
All BEA MessageQ API functions supported by the MVS Client are processed using the following sequence of events:

1. The application issues a BEA MessageQ function call to the MVS Client library.
2. The MVS Client library verifies the function call arguments and sends them, in a request, to the CLS.

3. When a request arrives, the CLS issues the corresponding BEA MessageQ function call on the server group.
4. When the BEA MessageQ function call on the server group completes, the CLS sends the return parameters and function status, in a response, back to the MVS Client that sent the request.
5. The MVS Client library waits for the response from the CLS. When the response arrives, the MVS Client library function call returns to the application with the return arguments and function status.

Figure 1-2 shows the communication layers used by a BEA MessageQ MVS Client application.

Figure 1-2 BEA MessageQ MVS Client Library Architecture



The BEA MessageQ MVS Client library allows only one BEA MessageQ function call from the same application to be in progress at a time, with the exceptions of `pams_exit` and `pams_detach_q`. For example, if an application calls `pams_get_msgw` and then calls `pams_put_msg` before the get message function call returns, the `pams_put_msg` function returns immediately with an error status value of `PAMS__PREVCALLBUSY`.

The BEA MessageQ MVS Client library allows multiple applications to connect to queues on the message queuing bus. The library creates a separate network connection to the CLS for each application using BEA MessageQ. The total number of applications that can connect to the message queuing bus is limited by the number of TCP/IP sessions. The Client library uses the Berkeley Socket API for network services.

Store-and-Forward Journaling

The MVS Client provides an SAF journal to store recoverable messages when the connection to CLS is not available. Local SAF journal processing is available when Message Recovery Services (MRS) are enabled in the MVS Client configuration.

When message recovery is enabled, the journal is turned on when the BEA MessageQ MVS Client application first initiates an attach operation. If the CLS is not available at the time of an attach, the operation completes with a return status of `PAMS__JOURNAL_ON`.

When the communications link to the designated server is down, and MRS is enabled, messages using the following delivery modes are written to a disk 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`

Messages sent with these recoverable delivery modes are written to the SAF file, which uses the VSAM KSDS format. A `DD` statement for `DMQSAF` must be included in the execution JCL and the name must be the same as the one created when the environment was configured. When the communications link to the designated server is restored, all journaled traffic is transmitted, maintaining FIFO order for recoverable messages.

When the journal is on, messages sent using a reliable delivery mode are saved to the journal. When the connection to the CLS is reestablished, all messages in the SAF journal are sent before new messages are processed.

Before you can use the MRS configuration option of the MVS Client, you must create the SAF journal. Refer to “Configuring Message Recovery Services” in Chapter 2, “Configuring the BEA MessageQ MVS Client,” for information on how to create the SAF journal.

2 Configuring the BEA MessageQ MVS Client

This chapter describes how to configure the BEA MessageQ MVS Client. It includes the following topics:

- Verifying Your Installation
- Creating MVS Client Datasets
- Configuring the Server Connection
- Configuring Logging
- Configuring Message Recovery Services
- Configuring Tracing
- Verifying the Configuration
- Testing the Configuration

Verifying Your Installation

Before you configure your MVS Client, make sure that the BEA MessageQ MVS Client software is properly installed and operational on your system. Refer to the *BEA MessageQ MVS Client Installation Guide* for MVS Client installation information.

After successful installation of the BEA MessageQ MVS Client, files provided on the distribution CD-ROM are installed in six partitioned datasets created on your MVS system. The files in the datasets are described in Table 2-1.

Table 2-1 MVS Client Datasets

Install CD /MVS5_0 root	MVS Partitioned Datasets	Dataset Description
./CNTL	DMQ.V5R0.CNTL	BEA MessageQ JCL library
./H	DMQ.V5R0.H	BEA MessageQ C include files
./X	DMQ.V5R0.X	BEA MessageQ sample C program source files
./COB	DMQ.V5R0.COB	BEA MessageQ COBOL Copy Books and sample programs
./LIB	DMQ.V5R0.LIB	BEA MessageQ object library for Language Environment Batch and CICS, Version 4.1
./PROC	DMQ.V5R0.PROC	BEA MessageQ example programs

An FTP directory listing on the system that was used to install the MVS Client should match the listing shown in Listing 2-1.

Listing 2-1 FTP Listing of the MVS Client Datasets

```
FTP> cd 'dmq.v5r0'
250 "'DMQ.V5R0.'" is working directory name prefix
FTP> dir
200 Port request OK.
125 List started OK.
Volume Unit    Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
TSO001 3380   1998/11/19  1   10  FB      80  6160  PO  CNTL
TSO001 3380   1998/11/19  1   10  FB      80  6160  PO  COB
TSO001 3380   1998/11/19  1   10  VB     259  3200  PO  H
TSO001 3380   1998/11/19  1  100  FB      80  6160  PO  LIB
TSO001 3380   1998/11/19  1  200  FB      80  3200  PO  PROC
TSO001 3380   1998/11/19  1   20  VB     259  3200  PO  X

250 List completed successfully.
FTP>
```

Note: We recommend making the BEA MessageQ MVS Client files read only. Be careful not to modify the original set of datasets that you copied to your system during installation. When following the configuration instructions provided in the rest of this chapter, modify only your own copy of the datasets. In addition, perform all compiling and linking of applications using BEA MessageQ in the user's area, not in the installation area.

Creating MVS Client Datasets

The BEA MessageQ MVS Client provides an object library (`DMQ.V5R0.LIB`) to support BEA MessageQ enabled applications. In addition, the media includes the following JCL procedures in the partitioned dataset `DMQ.V5R0.CNTL` (which are templates to be used in configuring and building your BEA MessageQ applications with the MVS Client):

- `DMQCOMP`—Sample JCL to compile the sample BEA MessageQ C programs
- `DMQCONF`—Configuration JCL
- `DMQJRNL`—SAF creation JCL

- DMQSAFD—Display SAF contents JCL
- DMQELOG—Error log creation JCL
- DMQLOG—Log file creation JCL
- DMQLINK—Sample JCL to pre-link and linkedit the BEA MessageQ sample C programs using DMQ.V5R0.LIB for MVS batch
- DMQCICS—Sample JCL to linkedit a CICS 4.1 program
- DMQSAMP—Sample JCL to compile, prelink, linkedit, and execute a sample program
- DMQRUN—Sample JCL to execute the sample programs
- DMQCOBOL—Sample JCL to compile, pre-link, and linkedit the BEA MessageQ sample COBOL program
- DMQCOBCI—Sample JCL to translate, compile, prelink, and linkedit the BEA MessageQ sample CICS COBOL program

Required Datasets

You must create the following four datasets for the application that uses the BEA MessageQ MVS Client:

- DMQINI—Client/server configuration file
- DMQELOG—Event Log
- DMQLOG—Trace Log
- DMQSAF—Store and forward file

These datasets are used to convey information to and from the BEA MessageQ MVS Client on behalf of your application. At run time, these files are associated with your job by JCL DDnames (as described in Table 2-2).

To create these four datasets, use the JCL templates provided during the installation (in DMQ.V5R0.CNTL), replacing the necessary volume and high-level qualifier (HLQ). In addition, you need to make specific configuration changes to the DMQCONF procedure

(which creates the DMQINI dataset) as described in the sections following Table 2-3. After you have made these edits to the JCL files, submit these four MVS batch jobs (DMQCONF, DMQELOG, DMQLOG, DMQJRNL) to create the associated datasets.

Refer to Table 2-2 for further information about how to create the datasets required to run BEA MessageQ MVS Client applications.

Table 2-2 Creating MVS Client Datasets

Dataset	Description	
Client Configuration File	This file contains all of the information needed to configure BEA MessageQ on the MVS Client.	
	Required MVS DDname:	DMQINI
	Default dataset name:	DMQ.V5R0.INI
	JCL for creating dataset:	DMQCONF in the dataset DMQ.V5R0.CNTL
	Modifications needed to JCL:	Modify the JCL to create a specific dataset name for the configuration file by replacing all instances of DMQ.V5R0.INI in the JCL along with VOLUMES and TSOUSR and recompile.
Client Log File	This file contains log messages about BEA MessageQ events.	
	Required MVS DDname:	DMQELOG
	Default dataset name:	DMQ.V5R0.ELOG
	JCL for creating dataset:	DMQELOG in the dataset DMQ.V5R0.CNTL
	Modifications needed to JCL:	Modify the JCL to create a specific dataset name for the error log file for each application by replacing all instances of DMQ.V5R0.ELOG in the JCL along with VOLUMES and TSOUSR and recompile.

Table 2-2 Creating MVS Client Datasets

Dataset	Description	
Client Message Tracing File	This file contains messages that were sent to or received by the MVS Client application.	
	Required MVS DDname:	`DMQLOG`
	Default dataset name:	`DMQ.V5R0.LOG`
	JCL for creating dataset:	`DMQLOG` in the dataset `DMQ.V5R0.CNTL`
	Modifications needed to JCL	Modify the JCL to create a specific dataset name for the message tracing file used for each application by replacing all instances of `DMQ.V5R0.LOG` in the JCL along with `VOLUMES` and `TSOUSR` and recompile.
Store and Forward Journal File	This file contains all of the recoverable messages that could not be delivered to the Client Library Server process in the BEA MessageQ Server group.	
	Required MVS DDname:	`DMQSAF`
	Default dataset name:	`DMQ.V5R0.SAF`
	JCL for creating dataset:	`DMQJRNL` in the dataset `DMQ.V5R0.CNTL`
	Modifications needed to JCL:	Modify the JCL to create a specific dataset name for the SAF journal file for each application by replacing all instances of `DMQ.V5R0.SAF` in the JCL along with `VOLUMES` and `TSOUSR` and rebuild.

Table 2-3 describes the configuration options for the MVS Client.

Table 2-3 BEA MessageQ MVS Client Configuration Options

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

To configure the BEA MessageQ MVS Client, use the MVS Client Configuration JCL, `DMQCONF` as shown in Listing 2-2. A copy of this JCL is located in `DMQ.V5R0.CNTL`. The configuration file is a VSAM ESDS. Edit the JCL file to add the appropriate settings and job card; then submit the job for processing.

Update and run this JCL whenever changes to the definitions of any of the following system configuration elements are needed.

- Default server—identifies the default BEA MessageQ server through which the MVS Client application will communicate.
- Failover server—identifies a failover server through which the Client can communicate if the default server system is unavailable.
- Logging—determines whether BEA MessageQ events and messages are logged to a file for troubleshooting purposes.
- Message Recovery Services—specifies the use of a Store and Forward file to store messages, which enables them to be sent at a later time if they cannot be delivered.
- Tracing—determines whether detailed system event level information is written to the event log file for troubleshooting purposes.

Listing 2-2 Configuration JCL DMQCONF

```
//*****************************************
//ALLOC  EXEC PGM=IDCAMS
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE (DMQ.V5R0.INI)
  DEFINE CLUSTER(                                    -
                NAME(DMQ.V5R0.INI)                   -
                VOLUMES(TSOUSR)                      -
                RECORD(132 132)                      -
                NONINDEXED                           -
                RECORDSIZE(132 132)                  -
                NONSPANNED                           -
                SHAREOPTIONS(2 3)                    -
                FREESPACE (20 0)                     -
                CISZ(1024)                           -
                SPEED                                -
                )                                    -
         DATA(NAME(DMQ.V5R0.INI.DATA))
/*
//*
//ALLOCP  EXEC PGM=IDCAMS
//*
//SYSPRINT    DD SYSOUT=*
//*
//OUTFILE DD DSN=DMQ.V5R0.INI,DISP=SHR
//INFILE  DD *
(Default Server)
TransportType=TCP/IP
Hostname=server_host
Endpoint=5000
ReconnectMsgInterval=0
ReconnectTimerInterval=0
(Failover)
EnableAutomaticFailover=0
TransportType=TCP/IP
Hostname=failover_host
Endpoint=5000

 (Logging)
ErrorEvents=1
SentMessages=0
ReceivedMessages=0
SentCICSMessages=0
ReceivedCICSMessages=0
```

```
(MRS)
Enabled=0
JournalPath=./
JournalFileSize=0
JournalCycle=0
JournalSizeFixed=0
PreAllocate=1
JournalBlockSize=0

(Trace)
PamsTrace=0
DmqclTrace=0

//SYSIN   DD *
   REPRO INFILE(INFILE) OUTFILE(OUTFILE)
//*
//ST1        EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//IN         DD DSN=DMQ.V5R0.INI,DISP=SHR
//SYSIN      DD *
       PRINT  -
         CHAR -
         INFILE(IN)
/*
//
```

Note: The BEA MessageQ configuration file entries are case insensitive. All records are translated to uppercase.

Configuring the Server Connection

This section describes how to configure the connection to the BEA MessageQ Client Library Server (CLS) on the BEA MessageQ message server system. Configuration of the server connection consists of the following two items:

- Configuring the default server (required)

  The default server is used for all connections to the message queuing bus. If you enable automatic reconnect, applications that are attempting to connect to a server (or lose a connection to the CLS) try 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.

- Configuring the automatic failover server (optional)

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

Configuring the Default Server

The default server identifies the BEA MessageQ server system for all connections to the message queuing bus. If you enable automatic reconnect, applications that are attempting to connect 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 MVS 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 SAF journal are sent to the CLS before new operations can be performed. For example, when a `pams_get_msg` operation triggers the reconnect threshold and a successful automatic reconnect and attach operation completes, the SAF journal is completely drained before the `pams_get_msg` function call returns. Table 2-4 shows the default server configuration options.

Table 2-4 Configuring the Default Server

Option	Description
`TransportType`	The network-level transport used to send messages to the BEA MessageQ CLS. The MVS Client supports TCP/IP only.
`Hostname`	The name of the host running the BEA MessageQ CLS. The `Hostname` must have a corresponding entry in the local `hosts` file.
`Endpoint`	The endpoint used by the BEA MessageQ CLS. For more information about specifying the endpoint, see the CLS section of the *Installation and Configuration Guide* for your server platform.
`ReconnectMsgInterval`	The number of message operations that occur before the server attempts to reconnect. If set to 0, automatic reconnect is not enabled. If set to greater than 0, automatic reconnect is enabled. The BEA MessageQ MVS attempts to reconnect to the server using the ReconnectMsgInterval option as the threshold for making a new connection attempt. Any messaging operation call, such as `pams_attach_q`, `pams_put_msg`, or `pams_get_msg`, increments the count used to determine when to attempt another reconnect. When the number of operations attempted reaches the ReconnectMsgInterval threshold, a reconnect attempt is made. 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.

Configuring the Automatic Failover Server

If the primary (default) server is not available and you have enabled automatic failover, the BEA MessageQ MVS Client provides the option of connecting to a failover CLS. The failover options are ignored if automatic reconnect to the default server is enabled.

When you enable automatic failover, an MVS 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 made only during a call to `pams_attach_q`.

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 MVS 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 MVS 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 MVS Client applications attach to a temporary queue and use a request/response style of messaging. These applications send requests to one or more servers that return responses to the queue address that sent the request. If failover occurs, the MVS Client application 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.

Automatic failover is not appropriate for all applications. When client applications 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 application is not attached are saved on that system. If the client reconnects to the same queue name or number, but in a different (failover) BEA MessageQ group, the recoverable messages in the BEA MessageQ group where the default CLS is located are not delivered to the new queue address used by the MVS Client.

On the other hand, the previous scenario could use failover by making the BEA MessageQ server group (and all disk-based queuing resources) also failover to another system so that messages previously sent to the MVS Client are received after a failover transition. Table 2-5 describes the automatic failover server configuration options.

Table 2-5 Configuring the Automatic Failover Server

Option	Description
`EnableAutomatic-Failover`	If set to "1", automatic failover is enabled. The failover server is used when the default server is not available and automatic failover is enabled. The Reconnect Message Interval option (see Table 2-4) must be greater than 0.
`TransportType`	The network-level transport used to send messages to the failover server. The MVS Client supports TCP/IP only.
`Hostname`	The name of the host running the BEA MessageQ CLS. The `Hostname` must have a corresponding entry in the local `hosts` file.
`Endpoint`	The endpoint used by the BEA MessageQ CLS. For more information, see the startup information for the CLS in the *BEA MessageQ Installation and Configuration Guide* for your server platform.

Configuring Logging

The MVS Client logging feature enables developers to:

- (MVS Batch) - Write information about BEA MessageQ system events to the `DMQELOG` file for troubleshooting purposes. Logging events is an effective way to monitor the run-time behavior of an application.
- (CICS) - Write BEA MessageQ messages to the Event Temporary Storage Queue (TSQ). Use the `CEBR` browse utility to browse the Event TSQ. Message logging allows you to obtain a complete history of the messaging activity of your application.

Table 2-6 describes the logging configuration options. Note that you must perform a `pams_attach_q` operation for any of the "Log Messages" options to take effect.

Table 2-6 Configuring Logging

Option	Description
ErrorEvents	If set to "1", logs error events to the DD statement DMQELOG. The default behavior is to log error events. If error event logging is enabled, when errors to the CLS occur, the DD name of the configuration file used at the time of the connection attempt is also logged. In MVS batch applications, error messages are also written to the DD statement (SYSOUT) of the executing application.
SentCICSMessages	If set to "1", sends a copy of BEA MessageQ messages sent by the CICS DMQTLOG Temporary Storage Queue (TSQ). (CICS only)
SentMessages	If set to "1", sends a copy of BEA MessageQ messages sent by the application to the DD statement DMQLOG message log file. In MVS batch applications, messages are also sent to the DD statement SYSOUT. When running under CICS, the messages are sent to the DMQLOG defined by the File Control Table (FCT).
ReceivedCICSMessages	If set to "1", sends a copy of BEA MessageQ messages received by the application to the CICS DMQTLOG Temporary Storage Queue. (CICS only)
ReceivedMessages	If set to "1", sends a copy of BEA MessageQ messages received by the application to the DD statement DMQLOG message log file. In MVS batch applications, messages are also sent to the DD statement SYSOUT. When running under CICS, the messages are sent to the DMQLOG defined by the File Control Table (FCT).

To perform error logging, an error log file must be created. To create an error log file, modify the DMQELOG JCL statement code that is illustrated in Listing 2-3 (supplied with the MVS Client) and submit the job for processing. The error log file must be VSAM ESDS. The template JCL is contained in DMQ.V5R0.CNTL as member DMQELOG. A DD card for DMQELOG must be included in your BEA MessageQ application execution JCL.

Listing 2-3 DMQELOG JCL Code

```
//*****************************************
//ALLOC  EXEC PGM=IDCAMS
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE (DMQ.V5R0.ELOG)
  DEFINE CLUSTER(                                  -
                NAME(DMQ.V5R0.ELOG)                -
                VOLUMES(TSOUSR)                    -
                RECORD(132 132)                    -
                NONINDEXED                         -
                RECORDSIZE(132 132)                -
                NONSPANNED                         -
                SHAREOPTIONS(2 3)                  -
                FREESPACE (20 0)                   -
                CISZ(1024)                         -
                SPEED                              -
                )                                  -
         DATA(NAME(DMQ.V5R0.ELOG.DATA))
/*
//*
//ALLOCP  EXEC PGM=IDCAMS
//*
//SYSPRINT    DD SYSOUT=*
//*
//OUTFILE DD DSN=DMQ.V5R0.ELOG,DISP=SHR
//INFILE  DD *
DMQ ERROR LOG
//SYSIN   DD *
   REPRO INFILE(INFILE) OUTFILE(OUTFILE)
//*
//ST1        EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//IN         DD DSN=DMQ.V5R0.ELOG,DISP=SHR
//SYSIN      DD *
       PRINT -
       CHAR  -
         INFILE(IN)
/*
//
```

To perform message logging, a log file must be created. To create a log file, modify the DMQLOG JCL statement that is illustrated in Listing 2-4 (supplied with the MVS Client) and submit the job for processing. The log file must be a VSAM ESDS. The template JCL is contained in DMQ.V5R0.CNTL as member DMQLOG. A DD card for the DMQLOG must be included in your BEA MessageQ application execution JCL.

Listing 2-4 DMQLOG JCL Code

```
//*****************************************
//ALLOC  EXEC PGM=IDCAMS
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE (DMQ.V5R0.LOG)
  DEFINE CLUSTER(                                    -
                NAME(DMQ.V5R0.LOG)                   -
                VOLUMES(TSOUSR)                      -
                RECORD(1000 500)                     -
                NONINDEXED                           -
                RECORDSIZE(132 400)                  -
                NONSPANNED                           -
                SHAREOPTIONS(2 3)                    -
                FREESPACE (20 0)                     -
                CISZ(1024)                           -
                SPEED                                -
                )                                    -
         DATA(NAME(DMQ.V5R0.LOG.DATA))
/*
//*
//ALLOCP  EXEC PGM=IDCAMS
//*
//SYSPRINT    DD SYSOUT=*
//*
//OUTFILE DD DSN=DMQ.V5R0.LOG,DISP=SHR
//INFILE  DD *
   DMQ LOG
//SYSIN   DD *
   REPRO INFILE(INFILE) OUTFILE(OUTFILE)
//*
//ST1        EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//IN         DD DSN=DMQ.V5R0.LOG,DISP=SHR
//SYSIN      DD *
       PRINT -
       CHAR  -
         INFILE(IN)
```

```
/*
//
```

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 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.

When message recovery is enabled, messages sent using the following reliable delivery modes are saved to the store and forward file:

- `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 re-established, 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 the 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 the SAF contains pending messages.
- 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 kept, pending transmission, in the SAF.

The MRS section of the initialization file is *optional* if recoverable messaging is not used. See Listing 2-5 for the MRS configuration options.

Listing 2-5 MRS Configuration Options

```
(MRS)
Enabled=0
JournalPath=./
JournalFileSize=0
JournalCycle=0
JournalSizeFixed=0
PreAllocate=1
JournalBlockSize=0
```

When the MRS Enabled field is set to zero, MRS is disabled. If the Enabled field is set to "1", message recovery services are enabled. The BEA MessageQ MVS Client uses a store and forward (SAF) file to store messages that cannot be delivered because a connection to the message queuing server is not available. When the connection to the server is restored, messages are delivered from the SAF file first; then normal messaging communication is restored.

If message recovery services are enabled in the initialization file, an SAF file must be created. An SAF file is a VSAM Key Sequenced Data Set (KSDS). The JCL to create this file is located in `DMQ.V5R0.CNTL`, as member `DMQJRNL`. A `DD` card for `DMQSAF` must be included in the BEA MessageQ application execution JCL.

To create an SAF file, modify the `DMQJRNL` JCL job statements illustrated in Listing 2-6 (supplied with the MVS Client) and submit the job for processing. The template JCL is contained in the `DMQ.V5R0.CNTL` as member `DMQJRNL`.

Listing 2-6 Create SAF File

```
//*****************************************
//ALLOC  EXEC PGM=IDCAMS
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DELETE (DMQ.V5R0.SAF)
  DEFINE CLUSTER(                                    -
                NAME(DMQ.V5R0.SAF)                   -
                VOLUMES(TSOUSR)                      -
                RECORD(500 100)                      -
                INDEXED KEY(4 0)                     -
                RECORDSIZE(2048 32000)               -
                UNIQUE NONSPANNED                    -
                SHAREOPTIONS(2 3)                    -
                FREESPACE (20 0)                     -
                CISZ(32000)                          -
                SPEED                                -
                )                                    -
         DATA(NAME(DMQ.V5R0.SAF.DATA))               -
         INDEX(NAME(DMQ.V5R0.SAF.INDEX))
/*
//*
//ALLOCP  EXEC PGM=IDCAMS
//*
//SYSPRINT    DD SYSOUT=*
//*
//OUTFILE DD DSN=DMQ.V5R0.SAF,DISP=SHR
//INFILE  DD *
000000000000000000000000000000000000000000DMQCL SAF V1.0
//SYSIN   DD *
   REPRO INFILE(INFILE) OUTFILE(OUTFILE)
//*
//ST1        EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//IN         DD DSN=DMQ.V5R0.SAF,DISP=SHR
//SYSIN      DD *
       PRINT -
       CHAR  -
         INFILE(IN)
/*
//
```

Configuring Tracing

Tracing can be a useful debugging tool, because it allows you to enable and disable BEA MessageQ MVS Client application processing activity trace output. The trace output may create large `SYSOUT` files on your system, and should be used only to monitor specific application behavior. Table 2-7 shows the tracing configuration options.

Table 2-7 Configuring Tracing

Option	Description
`PamsTrace`	If set to "1", logs the API calls to the file `DMQLOG`. The default is no tracing. On MVS batch, the messages are also directed to the `SYSPRINT DD` statement in the execution JCL.
`DmqclTrace`	If set to "1", traces the internal client library activity to the file `DMQLOG`. On MVS batch, the messages are also directed to the `SYSPRINT DD` statement in the execution JCL.

Verifying the Configuration

After you have successfully configured the BEA MessageQ MVS Client, you should have four VSAM datasets in the user area: `INI`, `LOG`, `ELOG`, and `SAF`. These datasets are referenced by the JCL that will run the BEA MessageQ application. An FTP directory listing from the system that was used to install the MVS Client should match the following listing.

Listing 2-7 FTP Listing of the MVS Client Configuration Datasets

```
FTP> dir
200 Port request OK.
125 List started OK.
Volume Unit Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
TSO001 3380                                     VSAM DMQ.ELOG
TSO001 3380                                     VSAM DMQ.ELOG.DATA
TSO001 3380                                     VSAM DMQ.INI
TSO001 3380                                     VSAM DMQ.INI.DATA
TSO001 3380                                     VSAM DMQ.LOG
TSO001 3380                                     VSAM DMQ.LOG.DATA
TSO001 3380                                     VSAM DMQ.SAF
TSO001 3380                                     VSAM DMQ.SAF.DATA
TSO001 3380                                     VSAM DMQ.SAF.INDEX

250 List completed successfully.
FTP>
```

Testing the Configuration

The MVS Client media contains sample programs to test your configuration. The sample programs are located in the `DMQ.V5R0.PROC` dataset. The C source code is located in the `DMQ.V5R0.X` dataset. The JCL to compile all of the sample C programs is member `DMQC370` in dataset `DMQ.V5R0.CNTL`. The JCL to linkedit all the sample C programs is member `DMQLINK` in dataset `DMQ.V5R0.CNTL`. `DMQLINK` requires a preallocated dataset for the linkage editor to store the executable programs. Please consult your systems programmer for the optimum block size for your disk configuration. The default name used by the supplied JCL procedures is `DMQ.V5R0.LINKLIB`. `DMQRUN` is located in `DMQ.V5R0.CNTL`. This JCL procedure executes the sample programs. Listing 2-8 lists the `DMQRUN` JCL.

Listing 2-8 The DMQRUN JCL File

```
//DMQRUN      JOB ,DMQ,CLASS=A,
//            MSGCLASS=X,MSGLEVEL=(1,1)
//            TIME=(5,00)
//*****************************************
//RUNIT  EXEC PGM=X@GET
//STEPLIB  DD DSN=DMQ.V5R0.PROC,DISP=SHR
//         DD DSN=CEE.V1R5M0.SCEERUN,DISP=SHR
//*
//DMQINI  DD DSN=DMQ.V5R0.INI,DISP=SHR
//DMQELOG DD DSN=DMQ.V5R0.ELOG,DISP=SHR
//DMQLOG  DD DSN=DMQ.V5R0.LOG,DISP=SHR
//DMQSAF  DD DSN=DMQ.V5R0.SAF,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD *
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.DATA
```

The `X@GET` example shown in Listing 2-8 attaches to a temporary queue on the server node, sends ten messages to itself, and receives them based on selection criteria. In addition, it sends ten more messages and receives them based on their priority setting. The C source statements for `X@GET` are located in `DMQ.V5R0.X`.

To run an MVS Client program, you must have a BEA MessageQ group, including a CLS running on a server system. No special group or queue definitions are required. Listing 2-9 shows the output of the `X@GET` sample program.

Listing 2-9 X@GET Program Output

```
X@Get Select Example starting...
Attached to queue:  ‘20, 1’
Putting messages:
  Put Message:
  Text:  ‘first message’
  Class:  FIRST
  Type:  ODD
  Getting FIRST messages
  Got message:
Class:  FIRST
  Type:  ODD
  Exited from MessageQ.
Pams_Get_msg Example complete....
```

3 Building Your Application

This chapter describes how to build your application, using the BEA MessageQ MVS Client library. Your system must have a program development environment that provides a compiler and linker. This chapter covers the following topics:

- BEA MessageQ API Support
- Sample Programs
- MVS Client-specific Return Codes
- Include Files for C and COBOL
- Compiling and Linking C Programs
- Compiling and Linking COBOL Programs
- Byte Order
- ASCII to EBCDIC Translation

BEA MessageQ API Support

The BEA MessageQ MVS Client supports most functions in the BEA MessageQ application programming interface (API) for the BEA MessageQ Server platforms. It does not support `pams_get_msga`, `pams_cancel_get`, `pams_open_jrn`, `pams_read_jrn`, and `pams_close_jrn`. To learn about the BEA MessageQ API calls, refer to the *BEA MessageQ Programmer's Guide*.

Sample Programs

The BEA MessageQ MVS Client is distributed with a number of sample application programs that demonstrate many of the features of the BEA MessageQ API. If the sample programs were installed during the MVS Client installation, they will be located in the following dataset.

`DMQ.V5R0.X`	C sample programs
`DMQ.V5R0.COB`	COBOL sample programs and Copy Books

The sample programs consist of C and COBOL language source programs. These samples are identical to the sample programs distributed with the other BEA MessageQ products, and demonstrate the portability of the BEA MessageQ API across all supported platforms.

To run the sample applications, you must define the server hostname and endpoint for your environment in the BEA MessageQ MVS Client configuration file. The Client Library Server (CLS) must be running on a BEA MessageQ Server for the sample program to work.

To build a sample program using TCP/IP to communicate with the server, refer to the following sections on linking and running BEA MessageQ applications for the MVS environment.

MVS Client-specific Return Codes

All BEA MessageQ return codes are defined as follows.

Return codes for . . .	Are defined in . . .
C programs	`P@RETURN` (an include file) in dataset `DMQ.V5R0.H`
COBOL programs	`DMQRC` (a COBOL Copy Book) in dataset `DMQ.V5R0.COB`

Some return codes are implemented only in the BEA MessageQ MVS Client; others are not implemented in the client.

Table 3-1 lists the return codes specific to the BEA MessageQ MVS Client.

Table 3-1 BEA MessageQ MVS Client-specific Return Codes

Return Code	Description
`PAMS__JOURNAL_FAIL`	MRS could not add messages to the local journal because of an operating system I/O error.
`PAMS__JOURNAL_FULL`	MRS 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 successfully wrote the message to the SAF journal. This return code can be treated as a success status.
`PAMS__LINK_UP`	The link to the CLS has been reestablished. This return code can be treated as a success status.
`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.

Include Files for C and COBOL

The BEA MessageQ MVS Client provides include files for C and COBOL programs. The include files contain the BEA MessageQ API function prototype declarations, BEA MessageQ return codes, symbolic constants used for API parameters, and additional declarations for using BEA MessageQ message-based services.

Table 3-2 lists the C language include files provided with the BEA MessageQ MVS Client. Note that the names of the MVS include files do not conform to the C language naming conventions due to MVS file naming restrictions. When referring to the include files in your C programs, use the C language file naming convention. For example, `P@ENTRY` should be included as `p_entry.h`.

Table 3-2 C Language Include Files

Include File	Contents
`P@ENTRY`	Function prototypes and type declarations for the BEA MessageQ API
`P@RETURN`	Return status values
`P@SYMBOL`	Symbolic constants used for function parameters
`P@PROCES`	Constant definitions for BEA MessageQ (for OpenVMS) processes
`P@GROUP`	Constant definitions for BEA MessageQ message-based services
`P@MSG`	PAMS message API structure definitions
`P@STATUS`	PAMS status message text translations
`P@TYPECL`	Constant definitions of BEA MessageQ message type and class for message-based services

The `P@RETURN` file defines the return codes using the values from the portable BEA MessageQ implementation for UNIX and Windows NT systems. When the CLS is running on OpenVMS, the return codes from BEA MessageQ for OpenVMS are converted to the equivalent values for the portable BEA MessageQ implementation.

Table 3-3 lists the BEA MessageQ MVS Client COBOL Copy Books and sample programs.

Table 3-3 COBOL Language Copy Books and Sample Programs

Copy Book	Contents
DMQCLASS	Definition of BEA MessageQ message classes
DMQGROUP	Definition of BEA MessageQ sample groups
DMQPCESS	Definition of BEA MessageQ sample queues
DMQRC	Definition of BEA MessageQ API return codes
DMQSAMP1	Sample COBOL batch program
ZDMQPROG	Sample COBOL CICS program

Compiling and Linking C Programs

The BEA MessageQ MVS Client contains a sample JCL to compile an MVS Client C program. To resolve the BEA MessageQ include files, you must specify the BEA MessageQ library DMQ.V5R0.H to the C compiler. Listing 3-1 shows the JCL for compiling MVS batch programs coded in C.

Listing 3-1 Sample JCL for Compiling MVS Batch C Programs

```
//COMP   EXEC EDCCLIB,
//       INFILE='DMQ.V5R0.X(userprog)',
//       LIBRARY='DMQ.USEROBJ',MEMBER='X@GET',
//   CPARM='LONGNAME,EXPMAC,NOMAR,DEF(MVS,IBM370),SOURCE,RENT'
//COMPILE.USERLIB DD DSN=DMQ.V5R0.H,DISP=SHR
```

If your application is going to be run in MVS batch, you must prelink with dataset DMQ.V5R0.LIB (specified in the SYSLIB DD statement in the prelink step). Listing 3-2 shows the JCL for linking for MVS batch programs coded in C.

Listing 3-2 Sample JCL for Linking MVS Batch C Programs

```
//*
//PLKED  EXEC EDCPL,
//   PPARM='MAP,NONCAL',
//   LPARM='LET,AMODE(31),RMODE(ANY),XREF,MAP',
//   INFILE=DMQ.V5R0.LIB(X@BASIC)',
//   OUTFILE='MQUSER.EXE(X@BASIC),DISP=SHR'
//PLKED.SYSLIB DD DSN=DMQ.V5R0.LIB,DISP=SHR
//SYSIN2  DD  *
   INCLUDE SYSLIB(CLNOCICS)
//LKED.SYSLIB  DD DSN=CEE.SCEELKED,DISP=SHR
//        DD   DSN=TCPIP.SEZACMTX,DISP=SHR
//        DD   DSN=TCPIP.SEZATCP,DISP=SHR
//*
```

Compiling and Linking COBOL Programs

To develop BEA MessageQ programs using COBOL, use the CLAPI programming interface. This interface converts all BEA MessageQ COBOL function calls into the equivalent C function calls documented in the *BEA MessageQ Programmer's Guide*.

In addition, because it is difficult to define a 1-byte integer using COBOL, the following BEA MessageQ arguments must be defined as `S9(9)COMP` in BEA MessageQ COBOL programs.

API Function	Argument name
`pams_confirm_msg`	`force_j`
`pams_get_msg`	`priority`
`pams_get_msgw`	`priority`
`pams_put_msg`	`priority`, `delivery`, and `uma`

The CLAPI interface adjusts the calling parameters from COBOL before making the BEA MessageQ function call. Refer to the sample COBOL programs for example field definitions. Table 3-4 lists the CLAPI function calls and their corresponding BEA MessageQ C program function calls.

Table 3-4 Supported COBOL Function Calls

BEA MessageQ CLAPI Function Call	BEA MessageQ C Function Call
CMQATTCH	pams_attach_q
CMQBIND	pams_bind_q
CMQLOCQ	pams_locate_q
CMQDETCH	pams_detach_q
CMQGET	pams_get_msg
CMQGETW	pams_get_msgw
CMQPUT	pams_put_msg
CMQCONF	pams_confirm_msg
CMQSSEL	pams_set_select
CMQCSEL	pams_cancel_select
CMQSTIME	pams_set_timer
CMQCTIME	pams_cancel_timer
CMQPEND	putil_show_pending
CMQILEA	putilea
CMQILAE	putilae
CMQSTSXT	pams_status_text
CMQEXIT	pams_exit

BEA MessageQ API arguments are called by reference, as follows:

```
CALL 'CMQATTCH'      USING   BY REFERENCE ATTACH-MODE
                             BY REFERENCE REQ-PROCESS-NUM
```

```
                                    BY REFERENCE Q-TYPE
                                    BY REFERENCE Q-NAME
                                    BY REFERENCE Q-NAME-LEN
                            RETURNING RETURN-CODE.
```

COBOL developers must be aware of the following points:

- All BEA MessageQ COBOL function calls are prefixed with `DMQ` instead of `CMQ`.
- You cannot use the `RETURNING` option of the `CALL` statement. The `PAMS` return code is stored in the COBOL reserved field, `RETURN_CODE`.
- You must check the API function status in `RETURN_CODE` to verify the result of your request.
- A sample COBOL program is located in `DMQ.V5R0.COB`.

The MVS Client includes a sample JCL to compile and link COBOL programs. The file is located in `DMQ.V5R0.CNTL` as member `DMQCOBOL`. Contact your systems programmer to obtain the COBOL library names used at your site. Listing 3-3 shows the sample JCL to compile and link BEA MessageQ COBOL programs.

Listing 3-3 JCL for Compiling and Linking COBOL Programs

```
//*
//   EXEC  IGYWCPL,
//   PGMLIB='MQUSER.EXE',
//   GOPGM=DMQSAMP1,
//   PARM.COBOL='LIB,APOST,OBJECT,MAP,LIST,XREF,NODYNAM,RENT',
//   PARM.LKED='AMODE(31),RMODE(ANY),XREF,MAP,LET=12'
//*
//SYSLIB DD  DSN=DMQ.V5R0.COB,DISP=SHR
//SYSIN  DD  DSN=DMQ.V5R0.COB(DMQSAMP1),DISP=SHR
//PLKED.SYSMSGS DD DSN=CEE.SCEEMSGP(EDCPMSGE),DISP=SHR
//PLKED.STEPLIB DD DSN=SYS1.SCEERUN,DISP=SHR
//PLKED.SYSLIB DD DSN=DMQ.V5R0.LIB,DISP=SHR
//SYSIN DD
//    DD  *
     INCLUDE SYSLIB(CLNOCICS)
//LKED.SYSLIB  DD DSN=CEE.SCEELKED,DISP=SHR
//        DD   DSN=TCPIP.SEZACMTX,DISP=SHR
//        DD   DSN=TCPIP.SEZATCP,DISP=SHR
//*
```

Byte Order

BEA MessageQ provides the capability to send and receive messages among 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). Application designers must take into account the differences in byte ordering when designing a distributed application with BEA MessageQ. IBM MVS systems use network byte order (big endian).

The byte order used on the BEA MessageQ MVS Client system and the CLS platform might be different. For example, a Digital Alpha system 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 MVS Client and CLS handle differences in byte ordering by using network byte order when the client and server systems 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 the CLS platform to initiate the messaging operation. However, the BEA MessageQ MVS Client does not perform byte-swapping on the user data passed in the message area for `pams_put_msg` or `pams_get_msg` calls.

Two alternative techniques are available for handling byte order differences in the client and server application components:

- You can send user data messages containing only character string data. Integer values are converted to the corresponding character representation before they are sent in a message.
- You can design an application-specific interface for sending and receiving messages that implements marshaling routines for the user data in each message.

For UNIX applications, data marshaling routines can be implemented as a set of library routines designed specifically to support data format conversion. These routines are typically written such that each marshaling routine performs one specific record conversion.

The Socket API provides routines to support byte-order conversion. These routines are `htonl`, `htons`, `ntohl`, and `ntohs`. For example, `htonl` means host to network long (32-bit) conversion. Marshaling routines can call these functions by including the file `socket.h`, and linking with the socket library.

ASCII to EBCDIC Translation

The IBM MVS environment uses the EBCDIC character encoding scheme. All platforms that support BEA MessageQ message servers use the ASCII character encoding scheme. The BEA MessageQ MVS Client provides two functions, `putilae` and `putilea`, for translating character data from ASCII to EBCDIC, and from EBCDIC to ASCII, respectively.

This section provides reference information for the `putilae` and `putilea` functions.

putilae

Translates character strings from ASCII to EBCDIC format. This function call is available on MVS systems only.

Syntax

```
int32 putilae (string, length)
```

Arguments

Argument	Data Type	Mechanism	Prototype	Access
`string`	`char`	reference	`char *`	passed/ returned
`length`	`short`	reference	`short *`	passed

Argument Definitions

`string`
Specifies the character string that is to be converted from ASCII to EBCDIC.

`length`
Specifies the length of the string to be converted.

Description

The `putilae` function accepts the string to be translated using the `string` argument and returns the translated string using the same argument.

Return Values

Return Code	Platform	Description
`PAMS__FAILED`	MVS	Unable to convert string
`PAMS__SUCCESS`	MVS	Normal successful completion

putilea

Translates character strings from EBCDIC to ASCII format. This function call is available on MVS systems only.

Syntax

```
int32 putilea (string, length)
```

Arguments

Argument	Data Type	Mechanism	Prototype	Access
`string`	`char`	reference	`char *`	passed/ returned
`length`	`short`	reference	`short *`	passed

Argument Definitions

`string`
Specifies the character string that is to be converted from EBCDIC to ASCII.

`length`
Specifies the length of the string to be converted.

Description

The `putilea` function accepts the string to be translated using the `string` argument and returns the translated string using the same argument.

Return Values

Return Code	Platform	Description
`PAMS__FAILED`	MVS	Unable to convert string
`PAMS__SUCCESS`	MVS	Normal successful completion

4 Running Your Application

This chapter explains how to run your application under both MVS batch and CICS. It includes the following sections:

- Running BEA MessageQ Applications Under MVS Batch
- Running BEA MessageQ Applications Under CICS

Running BEA MessageQ Applications Under MVS Batch

To use the BEA MessageQ MVS Client under MVS batch, the following software requirements must be met:

- Your BEA MessageQ Server and Client systems must have a network communication link. Have your TCP/IP network administrator verify the network connection to ensure that the systems are properly configured.
- The BEA MessageQ product for a UNIX, Windows NT, or OpenVMS platform must be installed on a BEA MessageQ Server system and a message queuing group must be configured to support the requirements of your messaging application environment. MVS Client applications use messaging resources, including message queues, message buffers, and system resources on the BEA MessageQ 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.

- For TCP/IP networking, 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 systems, as shown in the following table.

System	Location of Hosts File
UNIX	`/etc/hosts`
Windows NT	`c:\winnt\system32\drivers\etc\hosts`
OpenVMS	Use the TCP/IP for OpenVMS configuration utilities
MVS	Contact your TCP/IP administrator

For a complete description of BEA MessageQ Server systems and TCP/IP transports supported by the BEA MessageQ MVS Client, see the *BEA MessageQ MQSeries Connect and MVS Client Release Notes, Version 5.0.*

Several sample JCL execution files are included with the MVS Client software. Listing 4-1 shows sample JCL used to run the `X@GET` program in batch mode under MVS or OS/390. You can use this JCL to verify that your environment is properly configured.

Listing 4-1 Sample Execution JCL

```
//DMQRUN      JOB ,DMQ,CLASS=A,
//            MSGCLASS=X,MSGLEVEL=(1,1)
//            TIME=(5,00)
//*****************************************
//RUNIT  EXEC PGM=X@GET
//STEPLIB  DD DSN=DMQ.V5R0.PROC,DISP=SHR
//         DD DSN=CEE.V1R5M0.SCEERUN,DISP=SHR
//*
//DMQINI  DD DSN=DMQ.V5R0.INI,DISP=SHR
//DMQELOG DD DSN=DMQ.V5R0.ELOG,DISP=SHR
//DMQLOG  DD DSN=DMQ.V5R0.LOG,DISP=SHR
//DMQSAF  DD DSN=DMQ.V5R0.SAF,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD *
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.DATA
```

- Listing 4-2 includes sample JCL to run the COBOL sample program `DMQSAMP1` in batch mode under MVS or OS/390.

Listing 4-2 JCL for Executing COBOL Programs

```
//DMQSAMP1 EXEC PGM=DMQSAMP1
//STEPLIB  DD DSN=DMQ.V5R0.LINKLIB,DISP=SHR
//         DD DSN=CEE.V1R5M0.SCEERUN,DISP=SHR
//*
//DMQINI  DD DSN=DMQ.V5R0.INI,DISP=SHR
//DMQELOG DD DSN=DMQ.V5R0.ELOG,DISP=SHR
//DMQLOG  DD DSN=DMQ.V5R0.LOG,DISP=SHR
//DMQSAF  DD DSN=DMQ.V5R0.SAF,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTCPD  DD DSN=TCPIP.DATA,DISP=SHR
//DMQLIST  DD  SYSOUT=*,DCB=RECFM=FBA     (FOR COBOL PROGRAMS)
//*****************************************
```

Running BEA MessageQ Applications Under CICS

This section describes how to configure your environment to run BEA MessageQ applications under IBM CICS by:

- Defining FCT Entries
- Defining PCT Entries
- Using Message Tracing Under CICS
- Linkediting C Programs for CICS

Defining FCT Entries

To run BEA MessageQ applications in the MVS CICS environment, you must define FCT entries as shown in Listing 4-3.

Listing 4-3 FCT Entries to Define for a CICS Environment

```
DEFINE FILE(DMQINI) GROUP(DMQFILES)
  DSNAME(DMQ.V5R0.INI) LSRPOOL(1)
  DSNSHARING(ALLREQS) STRINGS(1)
  STATUS(ENABLED) OPENTIME(FIRSTREF) DISPOSITION(SHARE)
  DATABUFFERS(2) INDEXBUFFERS(1) RECORDFORMAT(V)
  ADD(YES) BROWSE(YES) DELETE(YES) READ(YES) UPDATE(YES)

DEFINE FILE(DMQELOG) GROUP(DMQFILES)
   DSNAME(DMQ.V5R0.ERR) LSRPOOLID(1)
   DSNSHARING(ALLREQS) STRINGS(1)
   STATUS(ENABLED) OPENTIME(FIRSTREF) DISPOSITION(SHARE)
   DATABUFFERS(2) INDEXBUFFERS(1) RECORDFORMAT(V)
   ADD(YES) BROWSE(YES) DELETE(YES) READ(YES) UPDATE(YES)

DEFINE FILE(DMQLOG) GROUP(DMQFILES)
  DSNAME(DMQ.V5R0.LOG) LSRPOOL(1)
  DSNSHARING(ALLREQS) STRINGS(1)
  STATUS(ENABLED) OPENTIME(FIRSTREF) DISPOSITION(SHARE)
  DATABUFFERS(2) INDEXBUFFERS(1) RECORDFORMAT(V)
  ADD(YES) BROWSE(YES) DELETE(YES) READ(YES) UPDATE(YES)

DEFINE FILE(DMQSAF) GROUP(DMQFILES)
  DSNAME(DMQ.V5R0.SAF) LSRPOOL(1)
  DSNSHARING(ALLREQS) STRINGS(1)
  STATUS(ENABLED) OPENTIME(FIRSTREF) DISPOSITION(SHARE)
  DATABUFFERS(2) INDEXBUFFERS(1) RECORDFORMAT(V)
  ADD(YES) BROWSE(YES) DELETE(YES) READ(YES) UPDATE(YES)
```

Defining PCT Entries

To run BEA MessageQ applications in the MVS CICS environment, you must define, to CICS, the PCT entries, the CICS transaction code, and the CICS BEA MessageQ program, as shown in Listing 4-4.

Listing 4-4 PCT Entries to Define for CICS Environment

```
DEFINE PROGRAM(X@GET) GROUP(DMQ)
  LANGUAGE(LE370) RELOAD(YES)
DEFINE TRANSACTION(XDMQ) GROUP(DMQ)
  PROGRAM(XDMQPROG) TWASIZE(1024) PROFILE(DFHCICST)
  TASKDATALOC(ANY)
```

Using Message Tracing Under CICS

If the trace options SENTCICSMESSAGES or RECEIVEDCICSMESSAGES are selected in the DMQ initialization file, messages are written to the transient data queue named DMQTLOG. The messages in this queue can be viewed using the CICS transaction CEBR. All records written to this queue are prefixed with the terminal ID, transaction code, date, and time.

Linkediting C Programs for CICS

The BEA MessageQ MVS Client contains a JCL sample for linkediting C programs in the CICS environment, as shown in Listing 4-5.

Listing 4-5 Sample Linkedit JCL for CICS C Programs

```
//DMQCICS JOB ,DMQ,CLASS=A,
//             MSGCLASS=X,MSGLEVEL=(1,1),
//             TIME=(5,00)
//*******************************************************
//PLINK   EXEC EDCPL,
//       INFILE='DMQ.USEROBJ(X@GET)',
//       OUTFILE='SYS2.USERLINK(XDMQPROG),DISP=(SHR,KEEP)',
//       PLIB='DSN=DMQ.V5R0.LIB,DISP=SHR',
//       PPARM='NONCAL',
//       LPARM='LET,AMODE(31),RMODE(ANY),XREF,LIST'
//LKED.SYSLIB  DD DSN=CEE.SCEELKED,DISP=SHR
//             DD DSN=TCPIP.V3R2.SEZATCP,DISP=SHR
//             DD DSN=TCPIP.V3R2.SEZACMTX,DISP=SHR
//             DD DSN=CICS410.SDFHLOAD,DISP=SHR
```

```
//LKED.SYSIN DD *
 INCLUDE SYSLIB(DFHELII)   REQUIRED BY CICS
 INCLUDE SYSLIB(EZACIC07)  REQUIRED BY CICS
//
```

5 Troubleshooting

This chapter describes how to identify and correct problems while running your application. The following topics are discussed:

- Run-time Errors
- Error Logging
- Failing to Connect to the CLS
- Network Errors
- TCP/IP Error Codes
- Tracing PAMS API Activity
- Tracing Client Library Activity
- Recovering from Client Crashes

Before you begin troubleshooting, you can verify the version number of the BEA MessageQ MVS Client library you are running by enabling tracing of Client library activity, running your application, and checking the error log file, `DMQELOG`, for the version number.

Run-time Errors

Problems at run time can arise from many circumstances. To identify and solve problems with the BEA MessageQ MVS Client, follow these steps:

1. Check the contents of the `DMQELOG` file to get more information about the problem. Network errors are identified in the error log file.

2. Use the trace output capability on the MVS Client and the Client Library Server (CLS) to get a detailed flow of the activity that leads up to the problem.
3. Log in to the BEA MessageQ Server system and run the BEA MessageQ Monitor Utility. On UNIX systems, use the character-cell program, `dmqmonc`, to monitor BEA MessageQ groups remotely.
4. Use the `netstat` TCP/IP utility to monitor the network connections on the client. Also, use `netstat` on the server system to monitor the TCP/IP connections on the host system where the CLS is running.
5. Try to repeat the error using one of the test programs included with the BEA MessageQ MVS Client. Reproducing problems with the test programs is an effective way to isolate application programming errors and it provides a convenient way to test problems.

Review the rest of this chapter for more information on finding and resolving problems.

Error Logging

Run-time errors detected by the Client library are written to the `DMQELOG` file. The errors indicate a run-time problem due to a configuration error, application error, network problem, or unexpected server response.

Error logging can be enabled or disabled by changing the BEA MessageQ MVS Client Configuration Logging option. Refer to Chapter 2, “Configuring the BEA MessageQ MVS Client,” for instructions on how to configure error logging.

Failing to Connect to the CLS

The BEA MessageQ MVS Client attempts to establish a connection to the CLS in response to a call to `pams_attach_q`. The attempt to establish a connection fails immediately when:

- There is no CLS running on the system identified in the BEA MessageQ MVS Client's default server configuration.
- The CLS is listening for connections on an endpoint other than the endpoint defined in the BEA MessageQ MVS Client's default server configuration.
- The CLS and the Client are using different network transports.

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

```
PAMS__NETNOLINK
```

Check the `DMQELOG` error file for the host name, and the endpoint of the server system with which the BEA MessageQ MVS Client attempted to connect. Then, either start the CLS on the server system, or reconfigure the BEA MessageQ MVS Client default server.

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
```

Network connection errors might also occur when you are 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
```

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

1. Check the error log file, `DMQELOG`, 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.

TCP/IP Error Codes

Table 5-1 lists the TCP/IP error status codes most frequently logged to the file DMQELOG when the MVS Client cannot connect to the CLS.

Table 5-1 Common TCP/IP Connect Errors

TCP/IP Error	Meaning	Recovery
EINVAL	A socket initialization error has occurred.	Verify the installation of TCP/IP MVS.
ECONNREFUSED	A connection request to the server is refused because a CLS is not listening on the endpoint.	Verify that the endpoint defined in the MVS Client/Server configuration endpoint matches the endpoint used by the CLS. Verify that the CLS is running.
ENOBUFS	TCP/IP 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 the steps outlined in the "Configuring Tracing" section in Chapter 2, "Configuring the BEA MessageQ MVS Client." The information from the pams_ function call trace is written to the dmqelog file in the

default directory for the application. The PAMS tracing option can be useful for observing 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, enable the `DmqclTrace` option in the client configuration file.

Recovering from Client Crashes

Occasionally applications crash (particularly during development) before closing or returning resources being used. Applications using the BEA MessageQ MVS Client attached to the message queuing bus that 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 “Network Errors” section of this chapter. 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 accepted only if the client application is attempting to reconnect from the same host used in 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 instance of the CLS remains active. For information about terminating CLS servers, see the "CLS" section in the *Installation and Configuration Guide* for your BEA MessageQ server platform.

Index