MQSeries® for Compaq OpenVMS Alpha®

Quick Beginnings

Version 5 Release 1

GC34-5885-00

MQSeries® for Compaq OpenVMS Alpha®

Quick Beginnings

Version 5 Release 1

GC34-5885-00

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix C. Notices" on page 85.

First edition (May 2001)

This edition applies to MQSeries for Compaq OpenVMS Alpha, Version 5.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Tables

Welcome to MQSeries for Compaq OpenVMS Alpha, V5.1

MQSeries for Compaq OpenVMS Alpha, V5.1—also referred to in this book as MQSeries® or MQSeries for Compaq OpenVMS—is part of the MQSeries family of products.

Note: *MQSeries for OpenVMS, Version 2* refers to MQSeries for Compaq (DIGITAL) OpenVMS, Versions 2.2.0, 2.2.1 and 2.2.1.1 unless explicitly stated otherwise.

This book is primarily for system administrators who manage the configuration and administration tasks for MQSeries. It describes MQSeries for Compaq OpenVMS and explains how to plan for and install the product. For detailed information about using MQSeries after it has been installed, refer to the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

How this book is organized

Use Table 1 to find the information you need to get started with MQSeries for Compaq OpenVMS.

Table 1. Getting Started Road Map

If you want to...	Refer to...
Learn about system requirements for installing MQSeries for Compaq OpenVMS	"Chapter 1. Planning to install an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 3
Install or migrate MQSeries for Compaq OpenVMS	"Chapter 2. Installing an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 9
Install or migrate an MQSeries client	"Chapter 4. Installing MQSeries for Compaq OpenVMS Alpha, V5.1 clients" on page 25
Learn about new features of MQSeries for Compaq OpenVMS V5.1	"What's new in MQSeries for Compaq OpenVMS Alpha, V5.1" on page ix
Read an introduction to MQSeries concepts	"Chapter 6. About MQSeries" on page 35
Start using command sets	"Chapter 7. Using MQSeries for Compaq OpenVMS" on page 45
View or print online documentation	"Chapter 8. Obtaining additional information" on page 65

Table 1. Getting Started Road Map (continued)

If you want to...	Refer to...
Contact IBM	See the *Readers comment form* at the back of the book

Conventions

Knowing the conventions used in this book will help you use it more efficiently.

- **Boldface type** indicates the name of an item you need to select or the name of a command.
- *Italics type* indicates new terms, book titles, or variable information that must be replaced by an actual value.
- `Monospace type` indicates an example (such as a fictitious path or file name) or text that is displayed on the screen.

What's new in MQSeries for Compaq OpenVMS Alpha, V5.1

This following new function is described in the current edition of the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

MQSeries queue manager clusters

MQSeries queue managers can be connected to form a cluster of queue managers. Within a cluster, queue managers can make the queues they host available to every other queue manager. Any queue manager can send a message to any other queue manager in the same cluster without the need for explicit channel definitions, remote queue definitions, or transmission queues for each destination. The main benefits of MQSeries clusters are:

- Fewer system administration tasks
- Increased availability
- Workload balancing

Note: MQSeries clusters are not the same as OpenVMS clusters. For a brief introduction to MQSeries queue manager clusters, see "Clusters" on page 39 and for more detailed information, see the *MQSeries Queue Manager Clusters* book. For more on how MQSeries works with Compaq OpenVMS clusters, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

MQSeries Application Interface (MQAI)

MQSeries for Compaq OpenVMS Alpha, V5.1 now supports the MQSeries Application Interface (MQAI), a programming interface that simplifies the use of PCF messages to configure MQSeries. For more information about the MQAI, including full command descriptions, see *MQSeries Administration Interface Programming Guide and Reference*.

Message queue size

A message queue can be up to 2 GB.

Controlled, synchronous shutdown of a queue manager

A new option has been added to the **endmqm** command to allow controlled, synchronous shutdown of a queue manager.

Java™ support

MQSeries for Compaq OpenVMS Alpha, V5.1 now works with Java compilers.

Web administration

With MQSeries for Compaq OpenVMS Alpha, V5.1, you can perform the following tasks using a Microsoft® Windows NT® system in association with an HTML browser, for example, Netscape Navigator or Microsoft Internet Explorer:

- Log on as an MQSeries Administrator
- Select a queue manager and issue MQSC commands against it
- Create, edit and delete MQSC scripts.

Part 1. Installing MQSeries for Compaq OpenVMS Alpha, V5.1

Chapter 1. Planning to install an MQSeries for Compaq OpenVMS Alpha, V5.1 server

This chapter summarizes the system requirements to run MQSeries, and the decisions you must make before installing MQSeries.

The following information applies to the server environment only. For information about installing a client, see "Chapter 4. Installing MQSeries for Compaq OpenVMS Alpha, V5.1 clients" on page 25.

Reading the release notes

Before installing MQSeries you are recommended to read the release notes for the product. These are included with the distribution kit and can be extracted before installation using the following command:

```
$ product extract release_notes mqseries/version=5.10/file=[mydir]myreleasenotes.txt
```

This command assumes your current directory is the same as the location for the Installation Kit. After the kit has been installed, you can find the release notes in: `sys$help:mqseries0510.release_notes`.

This file contains any additional information about MQSeries for Compaq OpenVMS Alpha, V5.1, including limitations, known problems and workarounds, and supersedes any corresponding information within this book.

Hardware requirements

MQSeries servers can be any Compaq Alpha system supported by the appropriate release of the OpenVMS operating environment, as shown in "Operating system requirements" on page 4.

Disk storage

A minimum of 50 MB (100,000 blocks) of disk space must be available for the product code and data on the server.

Note: Use the **show device** command to determine the amount of free space on your disk.

Hardware requirements

This is an approximate storage requirement for the installation. The installation requirements depend on which components you install and how much working space you need.

The usage space depends on the number of queues that you use, the number and size of the messages on the queues, and whether the messages are persistent or not. You may also require archiving capacity on disk, tape, or other media.

Working data for MQSeries for Compaq OpenVMS is stored by default in MQS_ROOT:[MQM].

Note: For added confidence in the integrity of your data, you are strongly advised to put your logs onto a *different* physical drive from the one that you use for the queues. This ensures that the size of the logs does not affect the space on the system disk or have an effect on performance.

Software requirements

For up-to-date information on supported software environments, refer to the MQSeries URL:

```
http://www.ibm.com/software/mqseries/platforms/supported.html
```

The system requires the following:

Operating system requirements

MQSeries for Compaq OpenVMS Alpha, V5.1 requires OpenVMS operating system V7.2–1 or V7.3.

Memory requirements

You are recommended to run MQSeries for Compaq OpenVMS Alpha, V5.1 on a system with a minimum of 128 MB of memory. Heavily loaded systems will benefit from additional memory.

Disk quotas

With the System Management utility (SYSMAN) supplied with OpenVMS, it is possible to enforce disk quotas for specific UICs on named disk volumes. If the MQS_ROOT:[MQM] directory is held on a volume that enables quota enforcement, you **must** also add the username MQM as an entry to the disk quota file.

First Failure Support Technology (FFST)® files contain important information used for MQSeries problem determination. When generated, these files are owned by MQM. It is therefore important to ensure that you allocate sufficient blocks to MQM if this feature is enabled on the volume. An

insufficient disk quota entry may lead to loss of FFST information and prevent timely resolution of MQSeries problems.

Connectivity

MQSeries for Compaq OpenVMS Alpha, V5.1 requires any communication hardware supporting DECnet or TCP/IP, or DIGITAL DECnet/SNA Gateway for Synchronous Transport.

For DECnet connectivity:

- DECnet-Plus for OpenVMS Version 7.2–1
- DECnet-Plus for Alpha Version 7.3

For TCP/IP connectivity:

- DIGITAL TCP/IP Services for OpenVMS AlphaV5.0a and V5.1, or
- Process Software TCPWare V5.4, or
- Process Software Multinet V4.3

For SNA connectivity: SNA APPC LU6.2 software and license must be installed. It must have access to a suitably configured SNA gateway.

- DECnet SNA Gateway ST V1.3, in conjunction with
- DECnet SNA LU6.2 API V2.4

Supported compilers

MQSeries for Compaq OpenVMS Alpha, V5.1 supports the following compilers:

- DEC C Version 6.2a
- DEC COBOL Version 5.7
- Java Version 1.1.8
- C++ Version 6.2

Options

You may use the following options with MQSeries for Compaq OpenVMS Alpha, V5.1.

Databases

- Oracle V8.1.6.0.0 (8iR2)

DCE

Compaq DCE for OpenVMS Alpha V3.0. This must be the U.S. Domestic version supporting DES encryption if you want to run the MQSeries-supplied DCE and send, receive or use message exits.

DCE names and security modules are provided with MQSeries for Compaq OpenVMS.

MQSeries for Compaq OpenVMS components

During the installation of MQSeries for Compaq OpenVMS Alpha, V5.1, you will be prompted to select which components you want to install.

The OpenVMS Server kit is called: `IBM-AXPVMS-MQSERIES-V0510--1.PCSI`. The components available in this kit are:

MQSeries Server:
MQSeries for Compaq OpenVMS Alpha, V5.1 server.

This comprises three interdependent components: MQSeries Server, MQSeries Base Kit for Client and Server, and MQSeries Runtime for Client and Server.

MQSeries Examples:
Sample MQSeries source code, including header files, link libraries and source files for sample applications. Samples are provided in C, C++ and COBOL.

MQSeries Java Client
Support for the Java client.

MQSeries Message Catalogs:
The US English message catalog is installed automatically and is always available. In addition to this, you may also choose to install support for MQSeries messages in any of the following languages:

- French
- German
- Italian
- Japanese
- Portuguese
- Spanish
- Korean
- Simplified Chinese
- Traditional Chinese

The OpenVMS Client kit is called `IBM-AXPVMS-MQCLIENT-V0510--1.PCSI`. The components available in this kit are:

MQSeries Client for OpenVMS:
MQSeries for Compaq OpenVMS Alpha, V5.1 client.

Notes:

1. Typically, a particular OpenVMS machine is designated as either an MQSeries client or a server, so you should install the corresponding MQSeries client or server component. However, the server component also

contains the full client content, so you can develop and use client-only applications on a machine that has the MQSeries server component installed without needing to install the client component.

2. Previous versions of MQSeries for Compaq OpenVMS shipped desktop client support for other platforms bundled with the OpenVMS Server installation kit. These clients are now packaged separately and may be found on the second distribution CD-ROM.

Things you need to know before installing

Before installing MQSeries for Compaq OpenVMS, you must:

- Extract and read the release notes as described in "Reading the release notes" on page 3.
- Be aware that the installation method has changed. Earlier versions of the product used the **VMSINSTAL** utility for installation. The current version has now been modified to use the Polycenter Software Installation Utility (PCSI), which is invoked using the operating system keyword **PRODUCT**. For further information on PCSI, see the online Help facility, specifying the keyword **PRODUCT**.
- Carry out the installation from the SYSTEM account as this will have the required privileges and quotas required for most product installations.
- Know the location of the software product kit. If the qualifier **/SOURCE** is not used, PCSI searches in the location defined by the logical PCSI$SOURCE for the installation kit. If this logical is not defined, the current directory is searched.
- The product files are installed in the default top-level directory for product files, which is: `SYS$SYSDEVICE:[VMS$COMMON]`.
- The installation creates an MQM account to be used by the server process. The default UIC value for this account is [400,400]. However, if the installation procedure detects that this UIC is already in use by another account on the system, you will be prompted to specify the next available UIC after [400,400], for example [400,401]. In addition, the installation also creates an MQS_SERVER account.
- To ensure proper security of the network and MQSeries, the MQM account **must** have a unique UIC. The password for this account is generated automatically and since this account is restricted, it is not necessary to know the account password. If this violates the security policies of your enterprise, you can modify the MQM account password using the OpenVMS **AUTHORIZE** utility after the installation has finished.

  Note: For the correct operation of MQSeries for Compaq OpenVMS, neither the MQM nor MQS_SERVER accounts should be removed.

Installation prerequisites

Change in client channel tables and how they may affect your installation

Important only if you are upgrading from an existing version to Version 5.1 and if you are currently using client channel tables

Before Version 5.1, the MQSeries for OpenVMS client and queue manager shared a client channel table file written in a format that is usable only on an OpenVMS system. As a result, the MQSeries for OpenVMS client could not read a client channel table file written by a non-OpenVMS queue manager, and a non-OpenVMS client could not read a client channel table file written by an MQSeries for OpenVMS queue manager.

With MQSeries for Compaq OpenVMS Alpha, V5.1, this restriction is lifted. The MQSeries for OpenVMS queue manager now writes client channel table files using the same format as all other MQSeries platforms and the MQSeries for OpenVMS client reads client channel tables in that format. Therefore, MQSeries for Compaq OpenVMS Alpha, V5.1 may freely exchange client channel table files with all other MQSeries platforms and versions except versions of MQSeries for OpenVMS before Version 5.1.

This has implications if you are upgrading an MQSeries for OpenVMS client or server, but not both. For example, if you upgrade the server to Version 5.1 but not the client, then the client channel table files written by the new server will no longer be readable by the old clients on OpenVMS. If you upgrade an OpenVMS client to Version 5.1, but not the server, then the Version 5.1 client will **not** be able to read the channel table files written by pre-Version 5.1 OpenVMS servers.

If you are currently using client channel tables, then it is highly recommended that you upgrade both the server and client to Version 5.1 to avoid any problems caused by incompatible client and server channel table files.

What to do next

When you have finished checking that your system meets the hardware, software and disk storage requirements, and you have completed the tasks listed in “Things you need to know before installing” on page 7, then:

- If you are installing MQSeries for Compaq OpenVMS Alpha, V5.1 on a new system, see “Chapter 2. Installing an MQSeries for Compaq OpenVMS Alpha, V5.1 server” on page 9 for the next step.
- If you already have MQSeries for Compaq (DIGITAL) OpenVMS Version 2 installed on your system and you want to upgrade it by installing the latest release, see“Migrating to MQSeries for Compaq OpenVMS Alpha, V5.1” on page 19 for the next step.
- For future reference, if you need to apply an update or PTF, refer to the release notes that accompany the CSD.

Chapter 2. Installing an MQSeries for Compaq OpenVMS Alpha, V5.1 server

This chapter explains how to install MQSeries for Compaq OpenVMS Alpha, V5.1 for the first time.

Note: If you already have MQSeries for Compaq (DIGITAL) OpenVMS Version 2 installed on the system, then follow the instructions in "Migrating to MQSeries for Compaq OpenVMS Alpha, V5.1" on page 19.

Before you begin

Before installing MQSeries for Compaq OpenVMS Alpha, V5.1, make sure your system meets all hardware, software and disk storage requirements. See "Chapter 1. Planning to install an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 3.

Use the Compaq OpenVMS **PCSI** utility. For further details see the *Compaq OpenVMS System Management Utilities Manual: M-Z*.

Installation procedure

The installation kit is supplied as a PCSI product installation kit named: `IBM-AXPVMS-MQSERIES-V0510--1.PCSI`.

From the Compaq OpenVMS command prompt, type:

```
$ PRODUCT INSTALL MQSERIES /VERSION=5.10/SOURCE=<cdrom>
```

where:

<cdrom> is the device and directory location of the installation kit.

Notes:

1. During the installation, you will be prompted to select which components you want to install.
2. Default responses are given in square parentheses [] at the end of each prompt. Press the `Return` key to accept the default, or type a new response to change the selection.

3. When you are requested to enter the destination for the MQSeries data files, give the location in the form of a device name or a device and directory. This location will be the value assigned to the MQS_ROOT logical. The device name may be a logical name. The default destination is SYS$COMMON.
4. Ensure that you have sufficient space on this device for your MQSeries data files. This will depend on the number and size of your queue and log files.

The following text shows an example of the output seen during an installation:

```
$ product install MQSERIES /version=5.10/source=DKA400

The following product has been selected:
    IBM AXPVMS MQSERIES V5.10      Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency requirements.

IBM AXPVMS MQSERIES V5.10 MQSeries for Compaq OpenVMS Alpha V5.10

    (C) Copyright IBM Corp. 1996, 2001 All Rights Reserved.

%MQSERIES-I-PRECONFIGURE, pre-configuration processing

        Please choose which of the following components to install:

         - all the MQSeries components
         - MQSeries Server (12796 blocks)
         - MQSeries Base Kit for Client and Server (1420 blocks)
         - MQSeries Runtime for Client and Server (31180 blocks)
         - MQSeries Examples (3772 blocks)
         - MQSeries Java Client (2300 blocks)
         - MQSeries Message Catalogs - French (776 blocks)
         - MQSeries Message Catalogs - German (792 blocks)
         - MQSeries Message Catalogs - Italian (776 blocks)
         - MQSeries Message Catalogs - Japanese (684 blocks)
         - MQSeries Message Catalogs - Korean (612 blocks)
         - MQSeries Message Catalogs - Portuguese (720 blocks)
         - MQSeries Message Catalogs - Spanish (776 blocks)
         - MQSeries Message Catalogs - Simplified Chinese (452 blocks)
         - MQSeries Message Catalogs - Traditional Chinese (488 blocks)

Do you want to install all the MQSeries components [N]?:
Do you want to install MQSeries Server (12796 blocks) [N]?: y
MQSeries Base Kit for Client and Server (1420 blocks) (required)
MQSeries Runtime for Client and Server (31180 blocks) (required)
Do you want to install MQSeries Examples (3772 blocks) [N]?: y
Do you want to install MQSeries Java Client (2300 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - French (776 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - German (792 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - Italian (776 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - Japanese (684 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - Korean (612 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - Portuguese (720 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - Spanish (776 blocks) [N]?:y
Do you want to install MQSeries Message Catalogs - Simplified Chinese (452 blocks) [N]?:
Do you want to install MQSeries Message Catalogs - Traditional Chinese (488 blocks) [N]?:
```

```
        The selections you have made are:

          - MQSeries Server (12796 blocks)
          - MQSeries Base Kit for Client and Server (1420 blocks) (required)
          - MQSeries Runtime for Client and Server (31180 blocks) (required)
          - MQSeries Examples (3772 blocks)
          - MQSeries Message Catalogs - Spanish (776 blocks)

Would you like to reselect your options [Y/N]:

Do you want to run the IVP after the installation [Y]?:

        ************************************************************
        Enter the destination device or directory for the MQSeries
        data files.  This value will be assigned to the MQS_ROOT
        logical.
        ************************************************************

Enter the destination for the MQSeries data files [SYS$COMMON]:
%UAF-I-RDBADDMSG, identifier MQM value %X8001001D added to rights database
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-RDBDONEMSG, rights database modified

        ************************************************************
        The installation procedure will create an account called MQM
        to run the MQSeries server processes.  The account will be
        created with the MQM resource identifier granted and the
        following privileges:
        TMPMBX,NETMBX,PRMGBL,SYSGBL

        You must specify a unique group UIC for this account in
        order to ensure proper security of the network.  The
        password for this account will be generated.  You do not
        need to know the password, since the account is disabled.
        If this scenario violates your security policies, you may
        change it after the installation has finished via the
        OpenVMS AUTHORIZE utility.
        ************************************************************

Enter the UIC of the new MQM account[400,400]?:
%UAF-I-RDBADDMSGU, identifier MQS_SERVER value [000400,000400] added to rights database
%UAF-I-ADDMSG, user record successfully added
%UAF-I-ADDMSG, user record successfully added
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBDONEMSG, rights database modified
%UAF-I-GRANTMSG, identifier MQM granted to MQS_SERVER
%UAF-I-GRANTMSG, identifier MQM granted to SYSTEM
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBNOMODS, no modifications made to rights database

%MQSERIES-I-PRECONFIGURE, pre-configuration terminated

* This product does not have any configuration options.

    You must install SNA LU6.2 Services to communicate over LU6.2

    Do you want to continue? [YES]

Execution phase starting ...

The following product will be installed to destination:
    IBM AXPVMS MQSERIES V5.10      DISK$ALPHASYS:[SYS0.SYSCOMMON.]

Portion done: 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%

%MQSERIES-I-POSTINSTALL, post-installation processing

        The following system parameter(s) are low.  Please increase these
        to the required value(s) before executing the MQSeries startup
```

Installation procedure

```
        command procedure.
        SYSGEN Parameter        Current Value           Required Value
        ----------------        -------------           --------------
        CHANNELCNT                      256                     1024

%MQSERIES-I-POSTINSTALL, post-installation terminated

....100%

The following product has been installed:
    IBM AXPVMS MQSERIES V5.10      Layered Product

%PCSI-I-IVPEXECUTE, executing test procedure for IBM AXPVMS MQSERIES V5.10 ...
***Creating the IVP queue manager
MQSeries queue manager created.
Creating or replacing default objects for ivp.
Default objects statistics : 29 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
***Starting the IVP queue manager
MQSeries queue manager 'ivp' started.
***Creating the IVP Test queue
0790997, 5724-A38 (C) Copyright IBM Corp. 1996, 2001  ALL RIGHTS RESERVED.
Starting MQSeries Commands.

AMQ8006: MQSeries queue created.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
***Writing to the IVP Test queue
Sample AMQSPUT0 start
target queue is testq
Sample AMQSPUT0 end
***Reading from the IVP Test queue
Sample AMQSGET0 start
message <This is an IVP test message being read from the test queue.>
no more messages
Sample AMQSGET0 end
***Ending the IVP queue manager
MQSeries queue manager ending.
MQSeries queue manager ending.
MQSeries queue manager ended.
***Deleting the IVP queue manager
MQSeries queue manager deleted.
***IVP Completed Successfully
%PCSI-I-IVPSUCCESS, test procedure completed successfully

IBM AXPVMS MQSERIES V5.10: MQSeries for Compaq OpenVMS Alpha V5.10

    Release notes are available in SYS$HELP:MQSERIES0510.RELEASE_NOTES

    Insert the following line in SYS$MANAGER:SYSTARTUP_VMS.COM:
        @sys$startup:mqs_startup.com
    Insert the following line in SYS$MANAGER:SYSHUTDWN.COM:
        @sys$manager:mqs_shutdown.com
```

Post-installation tasks

When you finished installing MQSeries for Compaq OpenVMS Alpha, V5.1, you can:

- Review the release notes for the product. These are placed in SYS$HELP by the installation procedure.
- Modify the system startup procedure to ensure that the MQSeries system logicals are defined and that all MQSeries shared libraries are installed

during system startup. See "Setting up MQSeries system logicals and install MQSeries shared libraries".

- Set up one or more separate MQSeries Administrator accounts. See "Setting up a separate MQSeries Administrator account".
- Create additional identifiers for groups that use MQSeries. See "Creating identifiers for groups that use MQSeries" on page 14.
- Make modifications to system resource parameters using the recommended system-supplied tool: AUTOGEN. See "Setting system parameters" on page 15.
- Set up a system-wide command file or the login files for all the users so that MQSeries commands can be invoked as if they were native DCL commands. See "Allowing users to invoke MQSeries commands from DCL" on page 18.
- Change the language MQSeries uses, if necessary. See "Setting the language for MQSeries for Compaq OpenVMS" on page 18.

Setting up MQSeries system logicals and install MQSeries shared libraries

The MQSeries environment is set up using the command procedure:

```
SYS$STARTUP:MQS_STARTUP.COM
```

This should be invoked when the machine is restarted to define the MQSeries system logicals and load all MQSeries shared libraries as known images.

The following command line should be added to the system startup command file SYS$MANAGER:SYSTARTUP_VMS.COM:

```
$ @SYS$STARTUP:MQS_STARTUP.COM
```

The following command line should be added to the system shutdown command file SYS$MANAGER:SYSHUTDWN.COM:

```
$  @SYS$MANAGER:MQS_SHUTDOWN.COM
```

Setting up a separate MQSeries Administrator account

MQSeries administration can be performed through the SYSTEM account on Compaq OpenVMS. The MQSeries installation procedure provides all required quotas and grants all required privileges to the SYSTEM account for this purpose.

However, rather than the VMS System Manager, you may want to have another person, or just a separate account, to administer MQSeries functions in your enterprise.

You must perform the following steps to set up the MQSeries Administrator account:

1. Use the Compaq OpenVMS **Authorize** utility (taking care to spell **Authorize** as it is shown here) to set up an interactive account as your MQSeries Administrator, with the identical privileges and quotas as the MQM account created by the installation procedure.

   Note: The account that you intend to use to administer MQSeries, and which you create to do this, does not require any special privileges other than those described in this section.

   In this example the name of the account is `MQADMIN`.
2. Grant the MQM identifier to your MQSeries Administrator account, `MQADMIN`, as follows:
   a. `$ RUN AUTHORIZE`
   b. `UAF> GRANT/IDENTIFIER/ATTRIBUTE=RESOURCE MQM MQADMIN`
   c. Exit authorize using `<Ctrl Z>`

Note: You can verify that you have set up the account correctly, using the command:

```
$  @SYS$MANAGER:MQS_CHECKADMIN
```

Creating identifiers for groups that use MQSeries

The MQM identifier is created during installation and essentially grants access to MQSeries administrative functions. If MQSeries security is being used, you will need to create additional identifiers to represent the groups of OpenVMS accounts that can be granted access to MQSeries objects. These identifiers will be granted to application groups using the OpenVMS `Authorize` utility.

See the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* for further information about using MQSeries security features.

For example, users whose OpenVMS accounts are in different UIC groups may want to share MQSeries resources such as queues. The users of these common queues may be granted the identifier called PAYROLL. To do this, you:

1. Add the PAYROLL identifier as a resource as follows:
   a. `$ RUN AUTHORIZE`
   b. `UAF> ADD/IDENTIFIER/ATTRIBUTE=RESOURCE PAYROLL`
   c. Exit authorize using `<Ctrl Z>`.
2. Grant the PAYROLL identifier to the desired user accounts (in this case, DOMESTIC and OVERSEAS) as follows:
   a. `$ RUN AUTHORIZE`
   b. `UAF> GRANT/IDENTIFIER PAYROLL DOMESTIC`
   c. `UAF> GRANT/IDENTIFIER PAYROLL OVERSEAS`
   d. Exit authorize using `<Ctrl Z>`.
3. Grant appropriate MQSeries authorizations for the grouped user accounts, using the **setmqaut** command, according to the capabilities required:

```
setmqaut -m qm0 -t qmgr -g payroll +connect
setmqaut -m qm0 -t queue -n 401k.q -g payroll +inq +put +get
```

   Use `+connect` to allow the user group to connect to a desired queue manager.

   Use `+inq`, `+put`, `+get` to allow the user group to inquire upon, put messages to, and get messages from a desired queue.

Note: For MQSeries to recognize any authorization changes made to accounts, you must logout all instances of the account that has been changed and restart the queue manager to reload the Object Authority Manager (OAM).

Setting system parameters

MQSeries for Compaq OpenVMS Alpha, V5.1 uses various system resources which are controlled by SYSGEN parameters. Insufficient quotas may result in unexpected errors.

In particular, you must have sufficient free global pagelets and global sections available. To install MQSeries, the recommended minimum amounts of these resources are:

```
GBLSECTIONS 100
GBLPAGES 40000
```

This should be sufficient to initialize the MQSeries environment and start a single queue manager with default settings during the installation verification stage.

Your runtime requirements, however depend on your MQSeries configuration and workload. We recommend that you use the AUTOGEN command procedure described in "Changing system parameter values with AUTOGEN" regularly to check that your system parameter settings are appropriate for the workload.

Notes:

1. SYSGEN parameters are system wide and apply to all processes that are running.
2. MQSeries makes particular use of CHANNELCNT, and consequently a minimum value of 1024 is recommended.

Changing system parameter values with AUTOGEN

The AUTOGEN command procedure (`SYS$UPDATE:AUTOGEN.COM`) supplied with OpenVMS, is the recommended method used to adjust system parameters according to the workload of your system.

AUTOGEN runs a number of ordered phases, each of which has a specific task. The parameters you specify when you invoke AUTOGEN determine which phases are run. There are also two modes of processing. If you specify FEEDBACK mode, AUTOGEN is able to size values based on actual workload figures collected and saved by OpenVMS. If you specify NOFEEDBACK this information is not used.

The file `SYS$SYSTEM:MODPARAMS.DAT` should be modified to control the size and limits of those system parameters which AUTOGEN adjusts. For example, the following two lines:

```
MIN_GBLSECTIONS = 900
ADD_GBLPAGES = 150
```

are used to set a minimum value for the system parameter GBLSECTIONS of 900; and to increment the current value of the system parameter GBLPAGES by 150 respectively. For further detailed information on the phases and processing modes of AUTOGEN, refer to the *Compaq OpenVMS System Management Utilities Reference Manual: A-L*.

Run AUTOGEN after making the appropriate changes to system parameter values within `SYS$SYSTEM:MODPARAMS.DAT`.

AUTOGEN can be invoked as follows:

```
@SYS$SYSTEM:AUTOGEN:<start-phase> <end-phase> <execution-mode>
```

The newly calculated parameter values come into effect after the next system reboot.

One suggested method for using AUTOGEN is to execute the required phases in two parts. For example:

```
$ @SYS$SYSTEM:AUTOGEN: savparams genparams feedback
$ @SYS$SYSTEM:AUTOGEN: setparams reboot feedback
```

In this example, the first processing of AUTOGEN will process dynamic workload figures and use these when calculating new system parameter values. The feedback information and newly calculated values are written to a text file (SYS$SYSTEM:AGEN$FEEDBACK.DAT) which should be reviewed for warnings before proceeding with the subsequent AUTOGEN processing.

When AUTOGEN is invoked on the second occasion, the newly calculated system parameters are written to the system parameter file (SYS$SYSTEM:ALPHAVMSSYS.PAR). The system is then automatically shutdown and rebooted with the new parameter values.

For further information regarding the performance tuning of MQSeries, refer to the relevant chapter of the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

You are recommended to run AUTOGEN initially on a weekly basis, to adjust system parameters, as the additional increase in system workload attributed to MQSeries will further use system resources.

If any required resource becomes exhausted, an FFST will be written detailing all relevant system and process quotas.

System limitations

MQSeries for Compaq OpenVMS Alpha, V5.1 has significant changes from the previous versions during startup. Each queue manager requires a minimum of six process slots, one for each of the processes created during the startup of the queue manager. These processes are:

Execution controller
Logger
Checkpointer
Repository manager
Channel initiator
Agent processes

Note that the agents are created multithreaded, supporting a maximum of 128 threads. Therefore, additional agent processes will be created when this limit is reached. The actual number of additional process slots required on the system depends on the number of queue managers created and the MQSeries workload. For heavy MQSeries workloads it may be necessary to increase the SYSGEN parameters `MAXPROCESSCNT` and `BALSETCNT`.

See the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* for details of trusted applications and performance tuning.

Setting the language for MQSeries for Compaq OpenVMS

Messages in US English are always available. If you require another of the languages that is supported by MQSeries for Compaq OpenVMS, you ***must*** ensure that your `SYS$NLSPATH` logical name includes the appropriate directory, and that you have installed the relevant MQSeries language component. This is normally done automatically by the MQSeries startup procedure. Furthermore, the `SYS$LC_ALL` logical name must specify the correct locale for the language, country and codeset.

For example, to select messages in German:

```
$ DEFINE/SYSTEM  SYS$LC_ALL  DE_DE_ISO8859-1.LOCALE
```

Allowing users to invoke MQSeries commands from DCL

MQSeries commands are implemented as DCL “foreign” commands. You should note that the DCL commands are not case sensitive.

To invoke MQSeries commands, which reside in the `SYS$SYSTEM` directory, as if they were native DCL commands, you **must** do the following.

Invoke the command file `SYS$MANAGER:MQS_SYMBOLS.COM` in the system-wide login file `SYS$MANAGER:SYLOGIN.COM`, or in the login files for all users who need to issue MQSeries commands.

Migrating to MQSeries for Compaq OpenVMS Alpha, V5.1

This section shows you how to migrate (upgrade) from MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1 (ECO8 or later) to MQSeries for Compaq OpenVMS Alpha, V5.1 running on OpenVMS V7.2–1. To migrate to the latest version, you:

- Perform the tasks described in "Before you begin". This includes stopping all queue managers, making sure the latest maintenance fix is installed on the system, and backing up the system.
- Remove from your system the existing version of MQSeries, install the latest version and then relink applications. This is described in "Migration procedure" on page 20.

When you are finished, you can check that the installation has worked properly by running the verification procedure.

Note: If you are installing MQSeries for Compaq OpenVMS Alpha, V5.1 on a system that does not have any of the previous versions of MQSeries installed, then use the procedure described in "Chapter 2. Installing an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 9.

Before you begin

Before you migrate from MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1 (ECO8 or later) to MQSeries for Compaq OpenVMS Alpha, V5.1, you must:

- Stop all queue managers. Use the **endmqm** command. See the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* for more information about the **endmqm** command.
- Check that your system meets all the requirements described in "Chapter 1. Planning to install an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 3, and that you have considered all the points outlined in "Things you need to know before installing" on page 7, and in particular, "Change in client channel tables and how they may affect your installation" on page 8.
- Make sure that you are currently running MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1 (ECO8 or later), that all your applications are working at this level and that you have installed the latest maintenance fix. If you are not sure what is running on your system, see "Querying the service level" on page 21 for guidance on how to find out.

  The latest maintenance fixes are available at: `http://www.ibm.com/software/mqseries/support/`. Follow the instructions on how to install the maintenance fix then migrate all applications to the new level.

- We strongly recommend that you take a backup copy of your system disk, and specifically a backup of the MQS_ROOT:[MQM] directory and its contents.

  Note: The MQS_ROOT:[MQM] directory on the disk represents that of the Version 2.2.1.1 product.

Migration procedure

To migrate from MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1 (ECO8 or later) to MQSeries for Compaq OpenVMS Alpha, V5.1, you must:

1. Remove the existing MQSeries from your system using the template file `MQS_CLEANOUT.TEMPLATE` which is supplied on the CD-ROM. To use the `MQS_CLEANOUT.TEMPLATE` file:
   a. Copy the file from the distribution media.
   b. Rename the template file to `.COM`.
   c. Run the resultant procedure.

   In the following example, the template command procedure is copied to `SYS$UPDATE`:

```
$ mount dka400: MQSERIES 510
$ copy/log dka400:[000000]mqs_cleanout.template   sys$update:mqs_cleanout.template
$ copy/log sys$update:mqs_cleanout.template  sys$update:mqs_cleanout.com
```

2. Remove the previous version of MQSeries by running the resultant command procedure as follows:

```
$ @sys$update:mqs_cleanout.com
```

   At this stage, MQSeries for Compaq (DIGITAL) OpenVMS, V2.2.1.1 (ECO8 or later) product files have been removed from the system. The user files (for example, the content of MQS_ROOT:[MQM]) and the existing MQSeries accounts and identifiers are preserved.

3. Now install MQSeries for Compaq OpenVMS Alpha, V5.1 using the PCSI installation method. For further details, refer to "Chapter 2. Installing an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 9.

   The directories and corresponding data belonging to existing Version 2 queue managers remain preserved in the MQS_ROOT:[MQM] directory structure, and will not be affected by the actual installation of the Version 5.1 product. Migration of this data to the Version 5.1 formats will occur automatically when these queue managers are first started using the Version 5.1 executables now installed.

4. Relink your applications.

Querying the service level

Note: You can use the following command to query the service level of MQSeries on your machine. However, the command is less likely to be used with MQSeries for Compaq OpenVMS Alpha, V5.1 because the PCSI utility contains extra functionality which supplies version information in a more explicit manner.

In general, for Version 2, to find out the current level of MQSeries installed on your OpenVMS system, use the following command:

```
$ analyze/system/inter sys$share:mqm.exe
```

Press the `Enter` key when prompted until you reach the screen containing the following information:

```
        Image identification information:



        image name:"MQM"
        image file identification:"MQS V2.211-009"
        image file build identification: ""
        link date/time: 10-MAR-2000 14:06:02.78
        linker identification:"A11-20"
```

The preceding example output displays the level installed as `V2.2.11-009`, which represents V2.2.1.1 (ECO9).

When MQSeries for Compaq OpenVMS Alpha, V5.1 is installed, you can use the following PCSI command to display the product information:

```
$ product show product mqseries
```

The output generated is as follows:

```
------------------------------------------------------------------------------
    PRODUCT                                KIT TYPE              STATE
------------------------------------------------------------------------------
    IBM AXPVMS MQSERIES V5.10              Full   LP             Installed
------------------------------------------------------------------------------
```

The preceding table shows that the current version of MQSeries installed on the system is V5.10. When the command is used with the /FULL qualifier, an additional column is generated in the output table corresponding to the level of the update kit added to the base product.

Restoring the previous backup version

If you experience difficulties with the new MQSeries V5.1 environment and your existing applications, you are recommended to revert to the previous version of MQSeries by restoring your previous backup. In addition, if the MQS_ROOT directory structure is on a device other than the system device then this must also be restored from backup to its original location.

For further information on the BACKUP command and its qualifiers, refer to the *System Management Utilities Reference Manual:A-L*.

If you do not have a satisfactory backup of your previous MQSeries working environment you should call your Customer Service Representative for further assistance.

Chapter 3. Verifying the installation for MQSeries for Compaq OpenVMS Alpha, V5.1

During the installation, you can choose to have MQSeries for Compaq OpenVMS Alpha, V5.1 automatically run an Installation Verification Program (IVP). If you have opted not to run the IVP during the installation process, or you have installed only one or two images as a result of a minor upgrade, you can use the following procedure to verify that the installation was successful. You are strongly recommended to test all the updated images to ensure that your new system runs as you expect.

Verifying the installation

Note: The installation procedure creates the MQM account and associated MQM resource identifier. If you are performing an upgrade the installation procedure detects that the account and identifier already exist and uses the existing values.

Follow these steps to verify your installation

You can invoke the IVP by running the following command procedure:

```
$  @sys$test:mqs_ivp.com
```

If problems occur, you may wish to run the IVP steps individually to isolate the cause. For a detailed description of the all the commands used in this procedure, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

To run the IVP steps individually:

1. Create a queue manager called `IVP`, by typing:

```
crtmqm IVP
```

Notes:

a. The queue manager name is not, in general, case sensitive. For more details on case sensitivity within OpenVMS, refer to the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

b. For the following steps, this example uses a queue manager called `IVP`.

2. Start the queue manager by typing:

```
strmqm IVP
```

   The **strmqm** command returns control when the queue manager has started and is ready to accept connect requests.
3. Create the `IVP` test queue called `testq` using the MQSC command **runmqsc**.
4. Write to the test queue using the supplied sample program in mqs_examples — AMQSPUT.
5. Write from the test queue using the sample program in mqs_examples — AMQSGET.
6. Stop the queue manager by typing:

```
endmqm IVP
```

7. Delete the queue manager by typing:

```
dltmqm IVP
```

   This command deletes the queue manager and its associated objects including the system default objects that you created in step 3.

Chapter 4. Installing MQSeries for Compaq OpenVMS Alpha, V5.1 clients

This chapter describes the system requirements to run a client for MQSeries and how to install a new client or upgrade an existing client.

The following information applies to the client environment only. For information about installing a server for MQSeries for Compaq OpenVMS, see "Chapter 1. Planning to install an MQSeries for Compaq OpenVMS Alpha, V5.1 server" on page 3.

Note: The new version of MQSeries for Compaq OpenVMS ships all the clients on a separate CD-ROM. Only the OpenVMS client is shipped on the server CD-ROM.

Read the release notes

The MQSeries Client kit contains release notes which describe further information that may supersede the information documented in this book. You are recommended to read through the release notes before installing the MQSeries Client for OpenVMS. The release notes can be obtained using the following command:

```
$ PRODUCT extract release_notes MQCLIENT/file=clientnotes.txt
```

In this example, the release notes for the MQCLIENT product are extracted and placed in a file called `clientnotes.txt` held in the current directory.

System requirements for MQSeries for Compaq OpenVMS clients

This section outlines the system requirements for an MQSeries for Compaq OpenVMS client.

Hardware

A Version 5.1 MQSeries client can run on any Alpha machine running OpenVMS Version 7.2–1 or Version 7.3. There must be enough random access memory (RAM) and disk storage for the programming prerequisites (below), the client code, the access methods and the application programs.

Disk storage

An MQSeries client requires 935 KB.

Compaq OpenVMS client requirements

Software

The following are required for MQSeries applications to run on a MQSeries for Compaq OpenVMS client.

- OpenVMS Version 7.2.1 or Version 7.3

Connectivity

MQSeries for Compaq OpenVMS Alpha, V5.1 requires any communication hardware supporting DECnet or TCP/IP, or DIGITAL DECnet/SNA Gateway for Synchronous Transport.

For DECnet connectivity:

- DECnet-Plus for OpenVMS Version V7.2–1
- DECnet-Plus for Alpha Version 7.3

For TCP/IP connectivity:

- DIGITAL TCP/IP Services for OpenVMS AlphaV5.0a and V5.1, or
- Process Software TCPWare V5.4, or
- Process Software Multinet V4.3

For SNA connectivity:

- DECnet SNA Gateway ST V1.3 in conjunction with
- DECnet SNA LU6.2 API V2.4

Compilers for MQSeries applications on Compaq OpenVMS Alpha clients

MQSeries for Compaq OpenVMS Alpha, V5.1 supports the following compilers:

- DEC C Version 6.2a
- DEC COBOL Version 5.7
- Java Version 1.1.8
- C++ Version 6.2

Components

MQSeries client
The MQSeries client code for your Compaq OpenVMS platform.

Samples
Sample application programs.

Support for DCE in Samples
This should be installed if you are going to use DCE.

Installing clients for MQSeries for Compaq OpenVMS V5.1

Before you install

Before you install an MQSeries for Compaq OpenVMS client on an Alpha machine, make sure your client machine meets all hardware, software and disk storage requirements for a client. See "System requirements for MQSeries for Compaq OpenVMS clients" on page 25.

Installation procedure

The installation kit is supplied as a PCSI installation kit named `IBM-AXPVMS-MQCLIENT-V0510-1.PCSI`.

From the Compaq OpenVMS command prompt, type:

```
$ PRODUCT INSTALL MQCLIENT/SOURCE=<cdrom>
```

where

<*cdrom*> is the device and directory location of the installation kit.

The following text is an example client installation script:

```
$ PRODUCT INSTALL MQCLIENT

The following product has been selected:
IBM AXPVMS MQCLIENT V5.10 Layered Product

Do you want to continue? [YES]

Configuration Phase Starting ...

You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.

IBM AXPVMS MQCLIENT V5.10:  IBM MQSeries Client for Compaq OpenVMS Alpha

   (C) Copyright IBM Corp. 1996, 2001 All Rights Reserved.

Do you want the defaults for all options? [YES]

Do you want to review the options? [NO] y

IBM AXPVMS MQCLIENT V5.10:  IBM MQSeries Client for Compaq OpenVMS Alpha
Compaq AXPVMS VMS V7.21 [Installed]
Do you wish to install the German message catalog?:  YES
Do you wish to install the Italian message catalog?:  YES
Do you wish to install the Korean message catalog?:  YES
Do you wish to install the Brazilian-Portuguese message catalog?:  YES
Do you wish to install the Spanish message catalog? : YES
Do you wish to install the French message?:  YES
Do you wish to install the Japanese message catalog?  YES
Do you wish to install the Simplified-Chinese message catalog?: YES
Do you wish to install the Traditional-Chinese message catalog?: YES
Do you wish to install the MQSeries help library?: YES

Are you satisfied with these options? [YES]

Execution phase starting ...

The following product will be installed to destination:
IBM AXPVMS MQCLIENT V5.10  DISK$SYSDSK0721:[VMS$COMMON.]

Portion done: 0% ... 10% ... 20%...30%...40%...50%...60%...70%...80%...90%...100%

The following product has been installed:
IBM AXPVMS MQCLIENT V5.10 Layered Product

IBM AXPVMS MQCLIENT V5.10:  IBM MQSeries Client for Compaq OpenVMS AXP
```

Note: Messages in U.S. English are always available. If you require messages to be displayed in one of the other supported language options, then the appropriate message catalog should be installed and the SYS$NLSPATH logical defined to include those messages from the appropriate directory.

Approximately 800 blocks are required to support each message catalog. The preceding example MQSeries client installation shows that *all* message catalogs have been selected. When the MQSeries client has been installed, the following record is added to the `VMSINSTAL.HISTORY` file:

```
------------------------------------------------------------
PRODUCT                          KIT TYPE STATE
------------------------------------------------------------
IBM AXPVMS MQCLIENT V5.10            Full LP Installed
------------------------------------------------------------
```

Migrating from an earlier version of a Compaq OpenVMS client

Use this section to migrate (or upgrade) an existing Compaq OpenVMS client to a Version 5.1 client.

Before you begin

Before you begin upgrading a client to Version 5.1:

- Make sure your client machine meets all hardware, software and disk storage requirements for a client. See "System requirements for MQSeries for Compaq OpenVMS clients" on page 25.
- If you have **not** upgraded the OpenVMS server to Version 5.1, and are planning to upgrade only the client, then see "Change in client channel tables and how they may affect your installation" on page 8 for some important information about potential non-compatibility issues between MQSeries for OpenVMS clients and MQSeries for OpenVMS servers that are not using the same version.
- You will have to relink applications when you are finished upgrading the client.

Migration procedure

To install the latest version of an MQSeries for OpenVMS client:

1. Log in as username SYSTEM.
2. To check whether the Version 2 client is installed, use the following command:

```
$ product show history mqseries
```

   If the client is installed you will see a result similar to the following:

```
------------------------------ ----------- ------------------------------------------
PRODUCT                        KIT TYPE          OPERATION              DATE AND TIME
------------------ ----------- ------------------------------------------------------
IBM AXPVMS MQSERIES V2.2       Full LP           Install        09-FEB-2000 15:46:09
-------------------------- ----------- ------------------------------------------
```

3. To remove the client, use the following command:

```
$ product remove  mqseries/version=2.2
```

4. Install the Version 5.1 MQCLIENT as described in "Installation procedure" on page 27.

   Note: The MQSeries Client Version 5.1 product name is MQCLIENT. The MQSeries Server Version 5.1 product name is MQSERIES.
5. If you used a client channel table file with the client before upgrading, then you must recreate the file using a MQSeries for Compaq OpenVMS Alpha, V5.1 queue manager (or a queue manager on any other platform).
6. Relink applications.

Chapter 5. Removing MQSeries

Before you remove MQSeries, do the following:

1. Ensure that you have stopped all MQSeries applications.
2. Ensure that you have stopped all the channels and that you have ended all the queue managers cleanly using the **endmqm** command.
3. Shutdown the MQSeries working environment by invoking the following, using the SYSTEM command:

```
$ @sys$manager:mqs_shutdown.com
```

4. If you are sure that you want to remove the MQSeries for Compaq OpenVMS Alpha, V5.1 product from your system, invoke the PCSI utility with the PRODUCT REMOVE command as follows:

```
$ PRODUCT REMOVE MQSERIES
```

Note: PCSI does **not** remove the MQS_ROOT:[MQM]directory structure.

In addition, the MQM and MQS_SERVER accounts and their corresponding identifiers also remain in the system authorization and rightslist files respectively. This action ensures that user applications and programs will still be operational when subsequent updates are applied, as the security profiles of these accounts, and associated application access control lists remain intact.

The following command shows how to remove MQSeries from the system disk:

```
$ PRODUCT REMOVE MQSERIES
The following product has been selected:
       IBM AXPVMS MQSERIES V5.10 Layered Product
Do you want to continue? [YES]
The following product will be removed from destination:
      IBM AXPVMS MQSERIES V5.10      DISK$SYSDSK0721:[SYS0.SYSCOMMON.]
Portion done:  0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
The following product has been removed:
      IBM AXPVMS MQSERIES V5.10 Layered Product
$
```

Part 2. Getting started with MQSeries

Chapter 6. About MQSeries

This chapter introduces IBM® MQSeries. It describes its basic functions and its relationships with operating systems, applications, and other middleware products. It contains the following sections:

Introduction

MQSeries is a communications system that provides assured, asynchronous, once-only delivery of data across a broad range of hardware and software platforms.

These characteristics make MQSeries the ideal infrastructure for application-to-application communication, and make it an appropriate solution whether the applications run on the same machine or on different machines that are separated by one or more networks.

MQSeries supports all the important communication protocols and even provides routes between networks that use different protocols. MQSeries bridges and gateway products allow easy access (with little or no programming) to many existing systems and application environments—for example, Lotus® Notes™, Web browsers, Java applets, and many others, although not all features are available on all platforms.

The assured delivery capability reflects the many functions built in to MQSeries to ensure that data is not lost because of failures in the underlying system or network infrastructure. Assured delivery enables MQSeries to form the backbone of critical communication systems and to be entrusted with delivering high-value data. There are also options that allow you to select a less robust quality of service, where this is appropriate. For example, there might be circumstances where you might prefer faster delivery with less emphasis on assured delivery.

The asynchronous processing support in MQSeries means that the exchange of data between the sending and receiving applications is time independent. This allows the sending and receiving applications to be decoupled so that the

sender can continue processing, without having to wait for the receiver to acknowledge that it has received the data. In fact, the target application does not even have to be running when the data is sent. Likewise, the entire network path between the sender and receiver may not need to be available when the data is in transit.

Once-only delivery of data is a vital consideration, particularly in financial and business applications where duplicate requests to move large sums of money from one account to another are precisely what you do not want to happen!

Messages, queues, and queue managers

The three fundamental concepts in MQSeries that you need to understand are:

- Messages
- Queues
- Queue managers

Messages

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

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

Queues

A *queue* is a data structure in which messages are stored. The messages may be put on, or got from, the queue by applications or by a queue manager as part of its normal operation.

Queues exist independently of the applications that use them. A queue can exist in main storage (if it is temporary), on disk or similar auxiliary storage (if it must be kept in case of recovery), or in both places (if it is currently being used, and must also be kept for recovery). Each queue belongs to a *queue manager*, which is responsible for maintaining it. The queue manager puts the messages it receives onto the appropriate queue.

Queues can exist either in your local system, in which case they are called *local queues*, or at another queue manager, in which case they are called *remote queues*.

Applications send to, and receive messages from, queues. For example, one application can put a message on a queue, and another application can get the message from the same queue.

Each queue has *queue attributes* that determine what happens when applications reference the queue. The attributes indicate:

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

Queue managers

A queue manager provides queuing services to applications, and manages the queues that belong to it. It ensures that:

- Object attributes are changed according to the details received.
- Special events (such as instrumentation events or triggering) are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a *local queue* to that queue manager. The queue manager to which an application is connected is said to be the local queue manager for that application. For the application, the queues that belong to its local queue manager are local queues. A *remote queue* is a queue that belongs to another queue manager. A *remote queue manager* is any queue manager other than the local queue manager. A remote queue manager may exist on a remote machine across the network or it may exist on the same machine as the local queue manager. MQSeries supports multiple queue managers on the same machine.

MQSeries configurations

In the simplest configurations, MQSeries is installed on a machine and a single queue manager is created. This queue manager then allows you to define queues. Local applications can then use these queues to exchange messages.

Communication by applications with queues managed by another queue manager requires *message channels* to be defined. It is not necessary to define a

channel directly to the target queue manager and it is often appropriate to define one only to the next hop (that is, an intermediate queue manager). Message channels available from that queue manager will be used to deliver the message to the target queue manager (or even to a subsequent hop).

More complex configurations can be created using a client-server structure. The MQSeries product can act as an MQSeries server to MQSeries clients. The clients and server do not need to be on the same platform. MQSeries supports a broad range of client platforms. The MQSeries products typically include clients for a variety of platforms. Additional MQSeries clients are available from the MQSeries Web site.

In a client-server configuration, the MQSeries server provides messaging and queuing services to the clients, as well as to any local applications. The clients are connected to the server through dedicated channels (known as *client channels*) for clients. This is a cost-effective deployment method because a server can support hundreds of clients with only a single copy of the MQSeries server product. However, the client channel must be continuously available whenever the MQSeries applications on the client are running. This contrasts with the message channels, which need not be continuously available to support MQSeries applications running on the server.

See “Channels” for more information.

MQSeries also supports the concept of MQSeries *clusters* to simplify setup and operation. An MQSeries cluster is a named collection of queue managers and any one queue manager can belong to none, one, or several such clusters. The queue managers in a cluster can exist on the same or different machines.

There are two major benefits from the use of MQSeries clusters:

1. Communication between members of a cluster is greatly simplified, particularly because the channels required for exchanging messages are automatically defined and created as needed.
2. Some or all of the queues of participating queue managers can be defined as being cluster queues, which has the effect of making them automatically known and available to all other queue managers in the cluster.

See “Clusters” on page 39 for more information.

Channels

A channel provides a communication path to a queue manager. There are two types of channel: message channels and MQI channels.

A *message channel* provides a communication path between two queue managers on the same, or different, platforms. The message channel is used for transmitting messages from one queue manager to another, and shields the

application programs from the complexities of the underlying networking protocols. A message channel can transmit messages in one direction only. Two message channels are required if two-way communication is required between two queue managers.

A *client channel* (also known as an *MQI channel*) connects an MQSeries client to a queue manager on a server machine and is bidirectional.

If you want to read more information about channels and how MQSeries uses them to communicate across the systems in your network, see the *MQSeries Intercommunication* book.

Clients and servers

MQSeries supports client-server configurations for MQSeries applications.

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

An *MQSeries server* is a queue manager that provides queuing services to one or more clients. All the MQSeries objects (for example, queues) exist only on the queue manager machine (that is, on the MQSeries server machine). A server can support local MQSeries applications as well.

The difference between an MQSeries server and an ordinary queue manager is that the MQSeries server can support MQSeries clients, and each MQSeries client application has a dedicated communication link with the MQSeries server.

For more information about client support, see the *MQSeries Clients* book.

Clusters

A cluster is a named collection of queue managers.

Note: Do not confuse MQSeries clusters with OpenVMS clusters. MQSeries Queue Manager Clusters do not make explicit use of OpenVMS Cluster intercommunication protocols, OpenVMS Cluster distributed lock manager and the OpenVMS Cluster file system.

See the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* for more information on MQSeries with Compaq OpenVMS clusters, and see the *MQSeries Queue Manager Clusters* book for more information on MQSeries clustering.

Clusters require that at least one of the queue managers in the cluster be defined as a *repository* (that is, a place where the shared cluster information can be held). More typically, two or more such repositories are usually designated to provide continued availability in the case of system failure. MQSeries makes sure that the information in the repositories is synchronized.

When a queue is defined as a cluster queue, it can be regarded as a public queue in that it is freely available to other queue managers in the cluster. This contrasts with non-cluster queues, which are accessible only when a local definition of them is available. Thus, a non-cluster queue has the characteristics of a private queue, accessible only to those queue managers that have been configured to know about them.

Public queues with the same name in the same cluster are regarded as equivalent. If a message is sent to that queue name, MQSeries (by default) sends it to any one of the instances, using a load-balancing algorithm. If you do not want this to happen, you can use the queue manager and queue name in the address, thus forcing the message to be delivered to a specific queue manager. Alternatively, you can replace the load-balancing routine with a different implementation. This is typical of MQSeries, in that there are many examples of where standard behavior can be changed by implementing user code in exits designed for this purpose.

You can read a full explanation in the *MQSeries Queue Manager Clusters* book.

MQSeries capabilities

MQSeries can be used to create many different types of solutions. Some exploit the platform support, or the bridge and gateway capabilities, to connect existing systems in an integrated way or to allow new applications to extract information from, or interchange information with, existing systems. Other solutions support business application servers, where a central pool of MQSeries applications can manage work sent across networks. Complex routing of information for workflow scenarios can be supported. Publish/subscribe or “send and forget” are other application scenarios that use different message flows. Load balancing and hot-standby systems can be built using the power and flexibility of MQSeries, which includes specific functions to support many of these diverse scenarios.

See the *MQSeries Application Programming Guide* for more information about writing MQSeries applications.

Transactional support

An application program may need to group a set of updates into a *unit of work*. Such updates are usually logically related and must all be successful for

data integrity to be preserved. Data integrity would be lost if one update in the group succeeded while another failed.

A unit of work *commits* when it completes successfully. At this point all updates made within that unit of work are made permanent and irreversible. Alternatively, all updates are *backed out* if the unit of work fails. *Syncpoint coordination* is the process by which a unit of work is either committed or backed out with integrity.

A *global* unit of work is one in which resources belonging to other resource managers, such as XA-compliant databases, are also updated. Here, a two-phase commit procedure must be used and the unit of work must be coordinated externally by another XA-compliant transaction manager such as IBM CICS®, IBM Transaction Server, IBM TXSeries™, Transarc Encina, or BEA Tuxedo.

The queue manager achieves this using a two-phase commit protocol. When a unit of work is to be committed, the queue manager first asks each participating database manager whether it is prepared to commit its updates. Only if all of the participants, including the queue manager itself, are prepared to commit, are all of the queue and database updates committed. If any participant cannot prepare its updates, the unit of work is backed out instead.

Full recovery is supported if the queue manager loses contact with any of the database managers during the commit protocol. If a database manager becomes unavailable while it is in doubt (that is, it has been called to prepare but has yet to receive a commit or backout decision), the queue manager remembers the outcome of the unit of work until it has been successfully delivered. Similarly, if the queue manager terminates with incomplete commit operations outstanding, these are remembered when the queue manager restarts.

Instrumentation events

You can use MQSeries instrumentation events to monitor the operation of queue managers.

Instrumentation events cause special messages, called *event messages*, to be generated whenever the queue manager detects a predefined set of conditions. For example, a *Queue Full* event message is generated if: Queue Full events are enabled for a specified queue; an application issues an MQPUT call to put a message on that queue; and the call fails because the queue is full.

Other conditions that can give rise to instrumentation events include:
- A predefined limit for the number of messages on a queue being reached
- A queue not being serviced within a specified time
- A channel instance being started or stopped

If you define your event queues as remote queues, you can put all the event queues on a single queue manager (for those nodes that support instrumentation events). You can then use the events generated to monitor a network of queue managers from a single node.

MQSeries instrumentation events are categorized as follows:

Queue manager events
These are related to the definitions of resources within queue managers. For example, if an application attempts to open a queue but the associated user ID is not authorized to perform that operation, a queue manager event is generated.

Performance events
These are notifications that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached or, following an MQGET request, a queue has not been serviced within a predefined period of time.

Channel events
These are reported by channels as a result of conditions detected during their operation. For example, a channel event is generated when a channel instance is stopped.

Message-driven processing

When they arrive on a queue, messages can automatically start an application, using a mechanism known as *triggering*. If necessary, the application can be stopped when the message or messages have been processed.

Programming MQSeries

MQSeries applications can be developed using a variety of programming languages and styles. Procedural and object-oriented programming is supported, depending on the MQSeries platform, using, for example, Visual Basic®, C, C++, Java, COBOL, PL/I, and TAL.

MQSeries function is logically divided into what is normally required by applications (such as putting messages on a queue) and what is necessary for administration (such as changing queue or queue manager definitions). Application function is known as the *MQI* (message queue interface). Administration function is known as the *MQAI* (message queuing administration interface). Applications can mix MQI and MQAI functionality, as required.

The administration functions can be implemented in two ways:

1. Most often, using MQAI language bindings or ActiveX™ classes.
2. Sending messages to administration queues, to achieve the same results as with the MQAI, using programmable command formats (PCFs).

Chapter 7. Using MQSeries for Compaq OpenVMS

This chapter introduces the command sets that can be used to perform system administration tasks on MQSeries objects. It covers:

Administration tasks include creating, starting, altering, viewing, stopping, and deleting MQSeries objects such as queue managers, queues, processes, channels, and namelists. To perform these tasks, you must select the appropriate command from one of the supplied command sets.

Introducing command sets

MQSeries provides three command sets for performing administration tasks:

- Control commands
- MQSC commands
- PCF commands

This section describes the command sets that are available. Some tasks can be performed using either a control command or an MQSC command, but other tasks can be performed using only one type of command. For a comparison of the facilities provided by the different types of command set, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

MQSeries command sets

This chapter introduces the MQSC, PCF, and control command sets, and provides a summary of the functions supported by each command set in the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*

Control commands

Control commands fall into three categories:

- *Queue manager commands*, including commands for creating, starting, stopping, and deleting queue managers and command servers.
- *Channel commands*, including commands for starting and ending channels and channel initiators.
- *Utility commands*, including commands associated with authority management and conversion exits.

Using control commands

MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide explains the syntax and purpose of each command.

You run control commands from the OpenVMS command prompt. Command names are not case sensitive.

The following list contains a brief description of each of the control commands. You can obtain help for the syntax of any of the commands by entering the command without any parameters. MQSeries responds by listing the syntax required for the selected command.

Command	Description
crtmqcvx	Creates a fragment of code that performs data conversion on data type structures.
crtmqm	Creates a local queue manager and defines the default and system objects.
dltmqm	Deletes a specified queue manager.
dmpmqlog	Dumps a formatted version of the MQSeries system log.
dspmqaut	Displays the current authorizations to a specified object.
dspmqcsv	Displays the status of the command server for the specified queue manager.
dspmqfls	Displays the real file system name for all MQSeries objects that match a specified criterion
dspmqtrc	Displays MQSeries formatted trace output.
dspmqtrn	Displays details of in-doubt transactions.
endmqcsv	Stops the command server on the specified queue manager.
endmqlsr	Ends a listener process.
endmqm	Stops a specified local queue manager.

Command	Description
endmqtrc	Ends tracing for the specified entity or all entities.
rcdmqimg	Writes an image of an MQSeries object, or group of objects, to the log for use in media recovery.
rcrmqobj	Recreates an object, or group of objects, from their images contained in the log.
rsvmqtrn	Commits or backs out internally or externally coordinated in-doubt transactions.
runmqchi	Runs a channel initiator process.
runmqchl	Runs either a Sender (SDR) or a Requester (RQSTR) channel.
runmqdlq	Starts the dead-letter queue (DLQ) handler, a utility that you can run to monitor and handle messages on a dead-letter queue.
runmqlsr	Runs a listener process.
runmqsc	Issues MQSC commands to a queue manager.
runmqtmc	Invokes a trigger monitor for a client.
runmqtrm	Invokes a trigger monitor.
setmqaut	Changes the authorizations to an object or to a class of objects.
strmqcsv	Starts the command server for the specified queue manager.
strmqm	Starts a local queue manager.
strmqtrc	Enables tracing.

MQSeries (MQSC) commands

You use the MQSeries (MQSC) commands to manage queue manager objects, including the queue manager itself, channels, queues, and process definitions. For example, there are commands to define, alter, display, and delete a specified queue.

When you display a queue, using the DISPLAY QUEUE command, you display the queue *attributes*. For example, the MAXMSGL attribute specifies the maximum length of a message that can be put on the queue. The command does not show you the messages on the queue.

For detailed information about each MQSC command, see the *MQSeries MQSC Command Reference* book.

Running MQSC commands

You run MQSC commands by invoking the control command **runmqsc**. You can run MQSC commands:

- Interactively by typing them at the keyboard
- As a sequence of commands from a text file

MQSeries command sets

For more information about using MQSC commands, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

PCF commands

MQSeries programmable command format (PCF) commands allow administration tasks to be programmed into an administration program. In this way you can create queues and process definitions, and change queue managers, from a program. PCF commands cover the same range of functions that are provided by the MQSC facility. You can therefore write a program to issue PCF commands to any queue manager in the network from a single node. In this way, you can both centralize and automate administration tasks.

Note: Unlike MQSC commands, PCF commands and their replies are not in a text format that you can read.

For a complete description of the PCF data structures and how to implement them, see the *MQSeries Programmable System Management* book.

Working with queue managers

This section describes how you can perform operations on queue managers, such as creating, starting, stopping, and deleting them. MQSeries provides control commands for performing these tasks.

Before you can do anything with messages and queues, you must create at least one queue manager.

Creating a queue manager

A queue manager manages the resources associated with it, in particular the queues that it owns. It provides queuing services to applications for Message Queuing Interface (MQI) calls and commands to create, modify, display, and delete MQSeries objects.

Before you can do anything with messages and queues, you must create at least one queue manager and its associated objects. To create a queue manager, you use the MQSeries control command **crtmqm**. The **crtmqm** command automatically creates the required default objects and system objects. Default objects form the basis of any object definitions that you make; system objects are required for queue manager operation. When a queue manager and its objects have been created, you use the **strmqm** command to start the queue manager.

Guidelines for creating queue managers

Before creating a queue manager, there are several points you need to consider (especially in a production environment). Work through this checklist:

- Specifying a unique queue manager name.
- Limiting the number of queue managers.
- Specifying a default queue manager.
- Specifying a dead-letter queue.
- Specifying a default transmission queue.
- Specifying the required logging parameters.
- Backing up configuration files after creating a queue manager.

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

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

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

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

Note: Queue manager names in control commands may or may not be converted to upper case, depending on an OpenVMS process option and whether the queue manager name was enclosed in double quotes to protect the case. This means that you could create two queue managers with the names `jupiter.queue.manager` and `JUPITER.queue.manager`. For more on how the OpenVMS process option and double quotes affect case, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

Limiting the number of queue managers: You can create as many queue managers as resources allow. However, because each queue manager requires its own resources, it is generally better to have one queue manager with 100 queues on a node than to have ten queue managers with ten queues each.

Creating queue managers

In production systems, many nodes will be run with a single queue manager, but larger server machines may run with multiple queue managers.

Specifying the default queue manager: Each node should have a default queue manager, though it is possible to configure MQSeries on a node without one.

To create a queue manager use the **crtmqm** command. For a detailed description of this command and its parameters, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

What is a default queue manager?

The default queue manager is the queue manager that applications connect to if they do not specify a queue manager name in an MQCONN call. It is also the queue manager that processes MQSC commands when you invoke the **runmqsc** command without specifying a queue manager name.

How do you specify a default queue manager?

You include the -q flag on the **crtmqm** command to specify that the queue manager you are creating is the default queue manager. Omit this flag if you do not want the queue manager you are creating to become a default queue manager.

Specifying a queue manager as the default *replaces* any existing default queue manager specification for the node.

What happens if I decide to change the default queue manager?

If you decide to change the default queue manager, be aware that this can affect other users or applications. The change has no effect on currently-connected applications, because they can use the handle from their original connect call in any further MQI calls. This handle ensures that the calls are directed to the same queue manager. Any applications connecting after the change connect to the new default queue manager.

This may be what you intend, but you should take this into account before you change the default.

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

Attention:

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

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

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

Therefore, when you create a queue manager you should use the -u flag to specify the name of the dead-letter queue. You can also use an MQSC command to alter the attributes of a queue manager and specify the dead-letter queue to be used.

When you find messages on a dead-letter queue, you can use the dead-letter queue handler, supplied with MQSeries, to process these messages. See the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* for further information about the dead-letter queue handler itself, and how to reduce the number of messages that might otherwise be placed on the dead-letter queue.

Specifying a default transmission queue: A transmission queue is a local queue on which messages in transit to a remote queue manager are queued pending transmission. The default transmission queue is the queue that is used when no transmission queue is explicitly defined. Each queue manager can be assigned a default transmission queue.

When you create a queue manager you should use the -d flag to specify the name of the default transmission queue. This does not actually create the queue; you have to do this explicitly later on. See the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* for more information.

Specifying the required logging parameters: You can specify logging parameters on the **crtmqm** command, including the type of logging, and the path and size of the log files. In a development environment, the default logging parameters should be adequate. However, you can change the defaults if, for example:

- You have a low-end system configuration that cannot support large logs.
- You anticipate a large number of long messages being on your queues at the same time.

Backing up configuration files after creating a queue manager: There are two configuration files to consider:

1. When you install the product, the MQSeries configuration file (mqs.ini) is created. It contains a list of queue managers, which is updated each time you create or delete a queue manager. There is one mqs.ini file per node.
2. When you create a new queue manager, a new queue manager configuration file (qm.ini) is automatically created. This contains configuration parameters for the queue manager.

You should make a backup of these files. If, later on, you create another queue manager that causes you problems, you can reinstate the backups when you have removed the source of the problem. As a general rule, you should back up your configuration files each time you create a new queue manager.

For more information about configuration files, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

Creating a default queue manager

You create a default queue manager using the **crtmqm** command. The **crtmqm** command specified with a q flag:

- Creates a default queue manager called `saturn.queue.manager`
- Creates the default and system objects
- Specifies the names of both its default transmission queue and its dead-letter queue

```
crtmqm -q -d MY.DEFAULT.XMIT.QUEUE -u SYSTEM.DEAD.LETTER.QUEUE "saturn.queue.manager"
```

where:

-q Indicates that this queue manager is the default queue manager.

-d MY.DEFAULT.XMIT.QUEUE
Is the name of the default transmission queue.

-u SYSTEM.DEAD.LETTER.QUEUE
Is the name of the dead-letter queue.

"saturn.queue.manager"
Is the name of this queue manager. For **crtmqm**, this must be the last parameter in the command.

Creating a default queue manager allows you to issue some commands against it (such as **strmqm** and **runmqsc**) without having to specify a queue manager name. Other commands (such as **endmqm** and **dltmqm**) require a specified queue manager name.

Notice that the queue manager name in this example is in lower case and that the lower case is protected by double quotes. For more information on how case sensitivity is handled for parameters, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* or "Specifying a unique queue manager name" on page 49.

Starting a queue manager

Although you have created a queue manager, it cannot process commands or MQI calls until it has been started. For example, to start a queue manager called `saturn.queue.manager`, type:

```
strmqm "saturn.queue.manager"
```

The **strmqm** command does not return control until the queue manager has started and is ready to accept connect requests.

Stopping a queue manager

You use the **endmqm** command to stop a queue manager. For example, to stop a queue manager, type:

```
endmqm "saturn.queue.manager"
```

Quiesced shutdown

By default, the **endmqm** command performs a *controlled* or *quiesced* shutdown of the specified queue manager. This may take a while to complete—a controlled shutdown waits until *all* connected applications have disconnected.

Use this type of shutdown to notify applications to stop. If you type:

```
endmqm -c "saturn.queue.manager"
```

you are not told when all applications have stopped. (An endmqm -c "saturn.queue.manager" command is equivalent to an endmqm "saturn.queue.manager" command.)

Immediate shutdown

For an immediate shutdown any current MQI calls are allowed to complete, but any new calls fail. This type of shutdown does not wait for applications to disconnect from the queue manager.

Use this as a normal way to stop the queue manager, optionally after a quiesce period. For an immediate shutdown, type:

```
endmqm -i "saturn.queue.manager"
```

Preemptive shutdown

Attention:
Do not use this method unless all other attempts to stop the queue manager using the **endmqm** command have failed. This method can have unpredictable consequences for connected applications.

If an immediate shutdown does not work, you must resort to a *preemptive* shutdown, specifying the -p flag. For example:

```
endmqm -p "saturn.queue.manager"
```

This stops all queue manager code immediately.

Note: After a forced or preemptive shutdown, or if the queue manager fails, the queue manager may have ended without cleaning up the shared memory that it owns. This can lead to problems restarting. For information on how to use the MONMQ utility to clean up after an abrupt ending of this type, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

If you have problems shutting down a queue manager

Problems in shutting down a queue manager are often caused by applications. For example, when applications:

- Do not check MQI return codes properly.
- Do not request a notification of a quiesce.
- Terminate without disconnecting from the queue manager (by issuing an MQDISC call).

If a shutdown of a queue manager is very slow, or you believe that the queue manager is not going to stop, you can break out of the **endmqm** command

using Ctrl-Y. You can then issue another **endmqm** command, but this time with a flag specifying either an immediate or a preemptive shutdown.

Restarting a queue manager

To restart a queue manager, use the command:

```
strmqm "saturn.queue.manager"
```

Deleting a queue manager

To delete a queue manager, first stop it, then use the following command:

```
dltmqm "saturn.queue.manager"
```

Attention:
Deleting a queue manager is a drastic step, because you also delete all the resources associated with it. This includes not only all queues and their messages, but also all object definitions. You should ensure that only trusted administrators have the authority to use this command.

Working with MQSeries objects

This section describes briefly how to use MQSC commands to create, display, change, copy, and delete MQSeries objects.

You can use the MQSC facility interactively (by entering commands at the keyboard) or you can redirect the standard input device (`SYS$INPUT`) to run a sequence of commands from a text file. The format of the commands is the same in both cases. The examples included here assume that you will be using the interactive method.

For more information about using MQSC commands, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

Before you can run MQSC commands, you must have created and started the queue manager that is going to run the commands.

Using the MQSC facility interactively

To start using the MQSC facility interactively, use the **runmqsc** command. Start an OpenVMS session and enter:

```
runmqsc
```

A queue manager name has not been specified; therefore the MQSC commands will be processed by the default queue manager. Now type in any MQSC commands, as required. For example:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

Continuation characters must be used to indicate that a command is continued on the following line:

- A minus sign (–) indicates that the command is to be continued from the start of the following line.
- A plus sign (+) indicates that the command is to be continued from the first nonblank character on the following line.

Command input terminates with the final character of a nonblank line that is not a continuation character. You can also terminate command input explicitly by entering a semicolon (;). (This is especially useful if you accidentally enter a continuation character at the end of the final line of command input.)

Feedback from MQSC commands

When you issue commands from the MQSC facility, the queue manager returns operator messages that confirm your actions or tell you about the errors you have made. For example:

```
AMQ8006: MQSeries queue created
 .
 .
 .
AMQ8405: Syntax error detected at or near end of command segment below:-
z

AMQ8426: Valid MQSC commands are:

    ALTER
    CLEAR
    DEFINE
    DELETE
    DISPLAY
    END
    PING
    REFRESH
    RESET
    RESOLVE
    RESUME
    START
    STOP
    SUSPEND
```

The first message confirms that a queue has been created; the second indicates that you have made a syntax error. These messages are sent to the standard output device. If you have not entered the command correctly, refer to the *MQSeries MQSC Command Reference* book for the correct syntax.

Ending interactive input to MQSC

To end interactive input of MQSC commands, type the MQSC END command:

```
END
```

Alternatively, you can exit by typing the EOF character <CTRL Z>.

If you are redirecting input from other sources, such as a text file, you do not have to do this.

Creating a local queue

For an application, the local queue manager is the queue manager to which the application is connected. Queues that are managed by the local queue manager are said to be local to that queue manager.

Use the MQSC command DEFINE QLOCAL to create a definition of a local queue and also to create the data structure that is called a queue. You can also modify the queue characteristics from those of the default local queue.

In this example, the queue we define, ORANGE.LOCAL.QUEUE, is specified to have these characteristics:

- It is enabled for gets, disabled for puts, and operates on a first-in-first-out (FIFO) basis.
- It is an 'ordinary' queue, that is, it is not an initiation queue or a transmission queue, and it does not generate trigger messages.
- The maximum queue depth is 1000 messages; the maximum message length is 2000 bytes.

The following MQSC command does this:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
       DESCR('Queue for messages from other systems') +
       PUT (DISABLED) +
       GET (ENABLED) +
       NOTRIGGER +
       MSGDLVSQ (FIFO) +
       MAXDEPTH (1000) +
       MAXMSGL (2000) +
       USAGE (NORMAL);
```

Notes:

1. Most of these attributes are the defaults as supplied with the product. However, they are shown here for purposes of illustration. You can omit them if you are sure that the defaults are what you want or have not been changed. See also "Displaying default object attributes".
2. USAGE (NORMAL) indicates that this queue is not an initiation queue or a transmission queue.
3. If you already have a local queue on the same queue manager with the name ORANGE.LOCAL.QUEUE, this command fails. Use the REPLACE attribute if you want to overwrite the existing definition of a queue, but see also "Changing local queue attributes" on page 60.

Displaying default object attributes

When you define an MQSeries object, it takes any attributes that you do not specify from the default object. For example, when you define a local queue, the queue inherits any attributes that you omit in the definition from the default local queue, which is called SYSTEM.DEFAULT.LOCAL.QUEUE. The default local queue is created automatically when you create the default

queue manager. To see exactly what these attributes are, use the following command:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE) ALL
```

Note: The syntax of this command is different from that of the corresponding **DEFINE** command.

You can selectively display attributes by specifying them individually. For example:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
        MAXDEPTH +
        MAXMSGL +
        CURDEPTH;
```

This command displays the three specified attributes as follows:

```
AMQ8409: Display Queue details.
    QUEUE(ORANGE.LOCAL.QUEUE)
    MAXDEPTH(1000)
    MAXMSGL(2000)
    CURDEPTH(0)
```

CURDEPTH is the current queue depth; that is, the number of messages on the queue. This is a useful attribute to display, because by monitoring the queue depth you can ensure that the queue does not become full.

Copying a local queue definition

You can copy a queue definition using the LIKE attribute on the **DEFINE** command.

For example:

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

This command creates a queue with the same attributes as our original queue ORANGE.LOCAL.QUEUE, rather than those of the system default local queue.

You can also use this form of the **DEFINE** command to copy a queue definition, but substituting one or more changes to the attributes of the original. For example:

```
DEFINE QLOCAL (THIRD.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE) +
       MAXMSGL(1024);
```

This command copies the attributes of the queue ORANGE.LOCAL.QUEUE to the queue THIRD.QUEUE, but specifies that the maximum message length on the new queue is to be 1024 bytes, rather than 2000.

Notes:

1. When you use the LIKE attribute on a **DEFINE** command, you are copying the queue attributes only. You are not copying the messages on the queue.
2. If you define a local queue, without specifying LIKE, it is the same as `DEFINE LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE).`

Changing local queue attributes

You can change queue attributes in two ways, using either the **ALTER QLOCAL** command or the **DEFINE QLOCAL** command with the REPLACE attribute. In "Creating a local queue" on page 57, we defined the queue ORANGE.LOCAL.QUEUE. Suppose, for example, you wanted to increase the maximum message length on this queue to 10 000 bytes.

- Using the **ALTER** command:
  This command changes a single attribute, that of the maximum message

```
ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
```

  length; all the other attributes remain the same.
- Using the **DEFINE** command with the REPLACE option, for example:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
```

  This command changes not only the maximum message length, but all the other attributes, which are given their default values. The queue is now put enabled whereas previously it was put inhibited. Put enabled is the default, as specified by the queue SYSTEM.DEFAULT.LOCAL.QUEUE, unless you have changed it.

If you decrease the maximum message length on an existing queue, existing messages are not affected. Any new messages, however, must meet the new criteria.

Deleting a local queue

Use the MQSC command **DELETE QLOCAL** to delete a local queue. A queue cannot be deleted if it has uncommitted messages on it. However, if the queue has one or more committed messages, and no uncommitted messages, it can be deleted only if you specify the PURGE option. For example:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Specifying NOPURGE instead of PURGE ensures that the queue is not deleted if it contains any committed messages.

Clearing a local queue

To delete all the messages from a local queue called MAGENTA.QUEUE, use the following command:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

You cannot clear a queue if:

- There are uncommitted messages that have been put on the queue under syncpoint.
- An application currently has the queue open.

Browsing queues

If you need to look at the contents of the messages on a queue, MQSeries for Compaq OpenVMS provides a sample queue browser for this purpose. The browser is supplied both as source and as a module that can be run. By default, the file names and paths are:

Source MQS_EXAMPLES:AMQSBCG0.C
Executable [.BIN]AMQSBCG.EXE, under MQS_EXAMPLES:.

The sample takes two parameters, which are the:

- Queue name, for example, SYSTEM.ADMIN.RESPQ.TEST.
- Queue manager name, for example, JJJH

as shown in the following command:

```
amqsbcg "SYSTEM.ADMIN.RESPQ.TEST" "JJJH"
```

There are no defaults; both parameters are required. Typical results from this command are shown below:

```
$ amqsbcg "SYSTEM.ADMIN.RESPQ.TEST" "JJJH"

 AMQSBCG0 - starts here
 **********************

 MQOPEN - 'SYSTEM.ADMIN.RESPQ.TEST'


 MQGET of message number 1
 ****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 819
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204A4A4A4820202020202020206EC8753A13200000'
 CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'JJJH                                            '
  ** Identity Context
  UserIdentifier : 'SYSTEM      '
  AccountingToken :
   X'0536353534300000000000000000000000000000000000000000000000000006'
  ApplIdentityData : '                                '
  ** Origin Context
  PutApplType    : '12'
  PutApplName    : 'AMQSPUT.EXE                '
  PutDate  : '20010129'    PutTime  : '19483901'
  ApplOriginData : '    '

  GroupId : X'000000000000000000000000000000000000000000000000'
  MsgSeqNumber   : '1'
  Offset         : '0'
  MsgFlags       : '0'
  OriginalLength : '14'

  ****   Message      ****
  length - 14 bytes

 00000000:  7465 7374 206D 6573 7361 6765 2031      'test message 1  '

  MQGET of message number 2
  ****Message descriptor****

  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 819
  Format : 'MQSTR   '
  Priority : 0  Persistence : 0
  MsgId : X'414D51204A4A4A4820202020202020206EC8753A23200000'
  CorrelId : X'000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'JJJH                                            '
```

```
 ** Identity Context
 UserIdentifier : 'SYSTEM
 'AccountingToken :
  X'0536353534300000000000000000000000000000000000000000000000000006'
 ApplIdentityData : '                                '
 ** Origin Context
 PutApplType    : '12'
 PutApplName    : 'AMQSPUT.EXE                '
 PutDate  : '20010129'    PutTime  : '19484323'
 ApplOriginData : '    '
   GroupId : X'000000000000000000000000000000000000000000000000'
 MsgSeqNumber   : '1'
 Offset         : '0'
 MsgFlags       : '0'
 OriginalLength : '14'

 ****   Message      ****

length - 14 bytes

00000000:  6D65 7373 6167 6520 3220 4441 5441      'message 2 DATA  '


 MQGET of message number 3
 ****Message descriptor****

 StrucId  : 'MD  '  Version : 2
 Report   : 0  MsgType : 8
 Expiry   : -1  Feedback : 0
 Encoding : 546  CodedCharSetId : 819
 Format : 'MQSTR   '
 Priority : 0  Persistence : 0
 MsgId : X'414D51204A4A4A4820202020202020206EC8753A33200000'
 CorrelId : X'000000000000000000000000000000000000000000000000'
 BackoutCount : 0
 ReplyToQ       : '                                               '
 ReplyToQMgr    : 'JJJH                                           '
 ** Identity Context
 UserIdentifier : 'SYSTEM      '
 AccountingToken :
 X'0536353534300000000000000000000000000000000000000000000000000006'
 ApplIdentityData : '                                '
 ** Origin Context
 PutApplType    : '12'
 PutApplName    : 'AMQSPUT.EXE                '
 PutDate  : '20010129'    PutTime  : '19491145'
 ApplOriginData : '    '
 GroupId : X'000000000000000000000000000000000000000000000000'
 MsgSeqNumber   : '1'
 Offset         : '0'
 MsgFlags       : '0'
 OriginalLength : '28'

****   Message      ****

length - 28 bytes

00000000:  6D65 7373 6167 6520 3320 6461 7461 202D 'message 3 data -'
00000010:  2065 6E64 206F 6620 696E 666F           ' end of info    '
  MQGET of message number 4
 ****Message descriptor****

 StrucId  : 'MD  '  Version : 2
 Report   : 0  MsgType : 8
 Expiry   : -1  Feedback : 0
 Encoding : 546  CodedCharSetId : 819
 Format : 'MQSTR   '
 Priority : 0  Persistence : 0
 MsgId : X'414D51204A4A4A4820202020202020206EC8753A43200000'
 CorrelId : X'000000000000000000000000000000000000000000000000'
```

```
BackoutCount : 0
ReplyToQ       : '                                                '
ReplyToQMgr    : 'JJJH                                            '
** Identity Context
UserIdentifier : 'SYSTEM      '
AccountingToken :
 X'0536353534300000000000000000000000000000000000000000000000000006'
ApplIdentityData : '                                    '
** Origin Context
PutApplType    : '12'
PutApplName    : 'AMQSPUT.EXE                '
PutDate  : '20010129'    PutTime  : '19510318'
ApplOriginData : '    '

GroupId : X'000000000000000000000000000000000000000000000000'
MsgSeqNumber   : '1'
Offset         : '0'
MsgFlags       : '0'
OriginalLength : '81'

****   Message      ****

length - 81 bytes

00000000:  4A4F 484E 534F 4E2C 4441 5649 4420 4D52 'JOHNSON,DAVID MR'
00000010:  2020 2020 3239 2D4A 414E 2D32 3030 3120 '    29-JAN-2001 '
00000020:  3133 3A34 3220 3431 3233 3030 3831 2031 '13:42 41230081 1'
00000030:  3238 332E 3334 2020 3030 3235 2E32 3220 '283.34  0025.22 '
00000040:  2030 3030 302E 3030 2020 3739 3235 2E36 ' 0000.00  7925.6'
00000050:  35                                      '5               '



No more messages
MQCLOSE
MQDISC
```

Chapter 8. Obtaining additional information

This chapter describes the documentation for MQSeries for Compaq OpenVMS Alpha, V5.1. It starts with a list of the publications, and then discusses:

- "Hardcopy Books"

MQSeries for Compaq OpenVMS Alpha, V5.1 is described in the following books:

Table 2. MQSeries books

Order Number	Title
MQSeries for Compaq OpenVMS Specific Books	
GC34–5885	*MQSeries for Compaq OpenVMS Alpha, V5.1 Quick Beginnings*
SC34–5884	*MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*
MQSeries Family Books	
SC33-1872	*MQSeries Intercommunication*
SC34-5349	*MQSeries Queue Manager Clusters*
GC33-1632	*MQSeries Clients*
SC33-1873	*MQSeries System Administration*
SC33-1369	*MQSeries MQSC Command Reference*
SC33-1482	*MQSeries Programmable System Management*
SC34-5390	*MQSeries Administration Interface Programming Guide and Reference*
GC33-1876	*MQSeries Messages*
SC33-0807	*MQSeries Application Programming Guide*
SC33-1673	*MQSeries Application Programming Reference*
SX33-6095	*MQSeries Programming Interfaces Reference Summary*
SC33-1877	*MQSeries Using C++*

Hardcopy Books

The book that you are reading now is *MQSeries for Compaq OpenVMS Alpha, V5.1 Quick Beginnings*. This book and the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide* are the only books that are supplied in hardcopy with the product. However, all books listed in Table 2 are available for you to order or print.

Hardcopy Books

You can order publications from the IBMLink™ Web site at:

```
http://www.ibm.com/ibmlink
```

In the United States, you can also order publications by dialing **1-800-879-2755**.

In Canada, you can order publications by dialing **1-800-IBM-4YOU (1-800-426-4968).**

For further information about ordering publications contact your IBM authorized dealer or marketing representative.

HTML and PDF Books on the World Wide Web

The MQSeries books are available on the World Wide Web as well as on the product CD-ROM. They are available in PDF and HTML format. The MQSeries product family Web site is at:

```
http://www.ibm.com/software/mqseries/
```

By following links from this Web site you can:
- Obtain latest information about the MQSeries product family.
- Access the MQSeries books in HTML and PDF formats.
- Download MQSeries SupportPacs.

You can access the Web versions of the books directly from the MQSeries Information Center (see the "Reference" section).

Online Help

Help pages are provided for all API calls, MQSC commands, and relevant control commands including **crtmqm**, **strmqm**, and **endmqm**.

Use the command:

```
$ HELP MQSERIES
```

Related publications

The following is a list of non-IBM publications that users of MQSeries for Compaq OpenVMS may find useful:

- *Compaq OpenVMS Performance Management*, January 1999

This book provides information to help you optimize performance on OpenVMS systems.

- *Compaq OpenVMS System Management Utilities* 2 volumes, January 1999

  These books contain reference information for system management utilities with OpenVMS.

- *Character Data Representation Library, Character Data Representation Architecture, Reference and Registry*, SC09–2190–00

  This document provides an overview of Character Data Representation Architecture (CDRA), and defines the elements of the architecture in the form of a reference manual.

- *DecNet SNA Gateway for Synchronous Transport Installation (OpenVMS)*, November 1993

  This guide explains how to install and configure DecNet SNA Gateway.
- *Digital SNA APPC/LU6.2 Programming Interface for OpenVMS*, May 1996

  This guide explains how to install and configure SNA APPC/LU6.2.
- *Digital TCP/IP Services for OpenVMS Installation and Configuration*, January 1999

  This guide provides instructions for installing and configuring Digital TCP/IP.
- *Guidelines for OpenVMS Cluster Configurations*, January 1999

  This guide describes how to maximize OpenVMS cluster availability and scalability.
- *Introduction to Compaq Networking and Data Communications*, (Compaq Part No. 093148)

  This guide provides an overview of Compaq networking and data communications concepts, tasks, products, and manuals.

Part 3. Appendixes

Appendix A. MQSeries for Compaq OpenVMS at a glance

Program and part number

- 5724–A38 MQSeries for Compaq OpenVMS, Version 5.1, part number 0790997.

Hardware requirements

MQSeries servers can be any Compaq Alpha machine with minimum of 128 MB of memory.

Software requirements

Software requirements are identical for server and client Compaq OpenVMS environments unless otherwise stated.

Minimum support levels are shown:

- Compaq OpenVMS Version 7.2–1 or Version 7.3

Connectivity

MQSeries for Compaq OpenVMS supports the following network protocols and hardware:

Network protocols:

- SNA LU6.2
- TCP/IP
- DECnet Phase V

And any communication hardware supporting DECnet or TCP/IP, or DIGITAL DECnet/SNA Gateway for Synchronous Transport.

For DECnet connectivity:

- DECnet-Plus for OpenVMS Version 7.2–1
- DECnet-Plus for Alpha Version 7.3

For TCP/IP connectivity:

- DIGITAL TCP/IP Services for OpenVMS AlphaV5.0a and V5.1, or
- Process Software TCPWare V5.4, or
- Process Software Multinet V4.3

Software requirements

For SNA connectivity: SNA APPC LU6.2 software and license must be installed. It must have access to a suitably configured SNA gateway.

- DECnet SNA Gateway ST V1.3, in conjunction with
- DECnet SNA LU6.2 API V2.4

Security

MQSeries for Compaq OpenVMS uses the security features of the Object Authority Manager (OAM) for MQSeries for Compaq OpenVMS.

All MQSeries resources run with the VMS Rights Identifier MQM. This rights identifier is created during MQSeries installation and you must grant this resource attribute to all users who need to control MQSeries resources.

Maintenance functions

MQSeries functions with:

- The **runmqsc** command-line interface.

Compatibility

The MQI for MQSeries for Compaq OpenVMS, Version 5 Release 1, is compatible with existing applications running MQSeries for Compaq (DIGITAL) OpenVMS, Version 2.2.1.1.

Supported compilers

Programs can be written using C, C++, COBOL, or Java.

- C programs can use the DEC C compiler
- C++ programs can use the DEC C++ compiler
- COBOL programs can use the DEC COBOL compiler
- Java programs

Language selection

A supplied message text file is encoded in the 7–bit character set that is native to the OpenVMS operating system.

Internationalization

MQSeries for Compaq OpenVMS lets the CCSID be specified when the queue manager instance is created. The queue manager CCSID defaults to 819. MQSeries for Compaq OpenVMS supports character-set conversion into the configured CCSID of the queue manager. For information about the CCSIDs that can be specified for an MQSeries for Compaq OpenVMS queue manager, including those that provide support for the euro character, see the *MQSeries Application Programming Reference* book.

Appendix B. Setting up communication in Compaq OpenVMS systems

Distributed queue management (DQM) is a remote queuing facility for MQSeries. It provides channel control programs for the queue manager that form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This appendix explains how to do this.

Deciding on a connection

There are three forms of communication for MQSeries for Compaq OpenVMS systems:

- TCP
- LU 6.2
- DECnet Phase V

Each channel definition must specify one only as the transmission protocol (Transport Type) attribute. One or more protocols may be used by a queue manager.

For MQSeries clients, it may be useful to have alternative channels using different transmission protocols. See the *MQSeries Clients* book.

Defining a TCP connection

The channel definition at the sending end specifies the address of the target. The TCP service is configured for the connection at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection Name field of the channel definition. Port number 1414 is assigned by the Internet Assigned Numbers Authority to MQSeries.

To use a port number other than the default, change the connection name field to the following:

```
Connection Name REMHOST(1822)
```

where *REMHOST* is the hostname of the remote machine and 1822 is the port number required. (This must be the port that the listener at the receiving end is listening on.)

Alternatively you can change the default sending port number by specifying it in the queue manager configuration file (qm.ini):

```
TCP:
   Port=1822
```

For more information about the values you set using qm.ini, see the *MQSeries for Compaq OpenVMS Alpha, V5.1 System Administration Guide*.

Using the TCP/IP SO_KEEPALIVE option

If you want to use the SO_KEEPALIVE option (as discussed in the *MQSeries Intercommunication* book) you must add the following entry to your queue manager configuration file (qm.ini):

```
TCP:
    KeepAlive=yes
```

Receiving end

There are two stages to defining the receiving end of a channel for MQSeries for Compaq OpenVMS. These are:

1. Defining a service using the TCP/IP stack installed on the system.
2. Starting the listener or receiver process once a message has been received by the TCP/IP service.

Defining a service using Digital TCP/IP Services for OpenVMS Alpha

To use Digital TCP/IP Services for OpenVMS Alpha, you must configure a TCP/IP service as follows:

1. Create a TCP/IP service to start the receiving channel program automatically:

```
$ TCPIP
     TCPIP> set service <p1>/port=<p2>/protocol=TCP -
     TCPIP> /user_name=MQM/process=<p3>/file=<p4>/limit=<p5>
```

   where:

   p1 Is the service name, for example MQSERIES01. A unique name is required for each queue manager defined.

   p2 Is the TCP/IP port number in the range 1 to 65 535. The default value for MQSeries is 1414.

   p3 Is the process name. This consists of a string up to 15 characters long.

   p4 Is the name of the startup command file that will be used to start the receiver, if it will be used; for example, `SYS$MANAGER:MQRECV.COM`. This is not required if the listener will be started using **runmqlsr**.

   p5 Is the process limit. This is the maximum number of connections allowed using the port number. If this limit is reached, subsequent requests are rejected.

   Note: Each channel represents a single connection to the queue manager.

2. To enable the service upon every system IPL (reboot), type the command:

```
$ TCPIP SET CONFIGURATION ENABLE SERVICE <p1>
```

3. To enable the service immediately (that is, without a system reboot), issue the command sequence:

```
$ TCPIP
TCPIP> enable service <p1>
```

Defining a service using Process Software MultiNet for OpenVMS

To use Process Software MultiNet for OpenVMS, you must configure a MultiNet service as follows:

1. Create a MultiNet service to start the receiving channel program automatically:

```
$ multinet configure/server
MultiNet Server Configuration Utility V3.5 (101)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
 SERVER-CONFIG> add <p1>
[Adding new configuration entry for service "MQSERIES"]
Protocol: [TCP]
TCP Port number: <p2>
Program to run: <p3>
[Added service MQSERIES to configuration]
[Selected service is now MQSERIES]
SERVER-CONFIG> set flags UCX_SERVER
[MQSERIES flags set to <UCX_SERVER>]
SERVER-CONFIG> set username MQM
[Username for service MQSERIES set to MQM]
SERVER-CONFIG> exit
[Writing configuration to MULTINET_COMMON_ROOT:SERVICES.MASTER_SERVER]
$
```

   where:

   p1 Is the service name, for example MQSERIES01. A unique name is required for each queue manager defined.

   p2 Is the TCP/IP port number in the range 1 to 65 535. The default value for MQSeries is 1414.

   p3 Is the name of the startup command file that will be used to start the receiver, if it will be used; for example, `SYS$MANAGER:MQRECV.COM`. This is not required if the listener is started using **runmqlsr**.

2. The service is enabled automatically after the next system IPL (reboot). To enable the service immediately, issue the command sequence:

```
$ multinet configure/server
MultiNet Server Configuration Utility V3.5 (101)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>restart
%RUN-S-PROC_ID, identification of created process is 0000021A
SERVER-CONFIG>exit
[Configuration not modified, so no update needed]
$
```

Defining a service using Process Software TCPWare

To use Process Software TCPWare, you must configure a TCPWare service as follows:

1. Edit the TCPWARE:SERVICES. file and add an entry for the service you want to use:

```
<p1> <p2>/tcp # MQSeries port
```

where:

p1 Is the service name, for example MQSERIES01. A unique name is required for each queue manager defined.

p2 Is the TCP/IP port number in the range 1 to 65 535. The default value for MQSeries is 1414. For example, an entry for a service MQSERIES01 on port 1414 would read:

```
MQSERIES01 1414/tcp # MQSeries port
```

2. Edit the TCPWARE:SERVERS.COM file and add an entry for the service defined in the previous step:

```
$! SERVERS.COM
$!
$ RUN TCPWARE:NETCU
NETCU> ADD SERVICE <p1> BG_TCP -
/INPUT=<p2> -
/LIMIT=<p3> -
/OPTION=KEEPALIVE -
/USERNAME=MQM
NETCU> EXIT
```

where:

p1 Is the service name, for example MQSERIES01. A unique name is required for each queue manager defined.

p2 Is the name of the startup command file that will be used to start the receiver, if it will be used; for example, SYS$MANAGER:MQRECV.COM. This is not required if the listener will be started using **runmqlsr**.

p3 Is the process limit. This is the maximum number of connections allowed using the port number. If this limit is reached, subsequent requests are rejected.

Note: Each channel represents a single connection to the queue manager.

3. The service is enabled automatically after the next system IPL. To enable the service immediately:

```
@TCPWARE:SERVERS.COM
```

Starting the listener or receiver process

There are two ways to start the receiver process with MQSeries for Compaq OpenVMS. These are:

1. Starting the receiver process using a command file to start the `amqcrsta` program.
2. Starting the listener process using the **runmqlsr** command.

It is only possible to use one method to start a specific port, but it is possible to start different ports on the same system using different methods. The TCP/IP service definitions described above may vary for each method, but it is possible to start the receiver using either method without changing the service definition.

Starting a receiver process using a command file

This was the only method of starting a receiver process under MQSeries for Compaq OpenVMS Version 2.2. This will cause a receiver process to be started when a message is first received on the port. There will be one receiver process for each receiver channel that is connected via this port.

1. Create a file consisting of one line and containing the DCL command to start the TCP receiver program amqcrsta.exe:

```
$ mcr amqcrsta [-m QMgrName]
```

   Place this file in the SYS$MANAGER directory. The name of the file must be the same as the startup command file defined in the service definition - in the examples above MQRECV.COM.
2. Ensure that the protection on the file and its parent directory allow it to be executable, that is, the protection is /PROT=W:RE.

Starting a listener process using the runmqlsr command

The **runmqlsr** command starts a listener process, regardless of whether a message has been received on the specified port. This process will listen on the specified port for incoming messages and will handle them as they arrive. Each server and receiver channel requires its own listener process. The format of the command is:

```
$ runmqlsr -t tcp [-p Port] [-m QMgrName]
```

If this method of starting a listener is chosen, there is no need to have a startup command file in the service definition. However, the presence of a startup command file will not cause any problems to the listener process.

Defining a DECnet Phase V connection

Set up the MQSeries configuration for channel objects:

1. Start the NCL configuration interface by typing the following command:

```
$ MC NCL
NCL>
```

2. Create a session control application entity by issuing the following commands:

```
NCL> create session control application MQSERIES
NCL> set sess con app MQSERIES address {name=MQSERIES}
NCL> set sess con app MQSERIES image name -
_ SYS$MANAGER:MQRECVDECNET.COM
NCL> set sess con app MQSERIES user name "MQM"
NCL> set sess con app MQSERIES node synonym true
NCL> show sess con app MQSERIES all [characteristics]
```

   Note: User-defined values are in upper case.

3. Create a file consisting of one line and containing the DCL command to start the DECnet receiver program, `amqcrsta.exe`:

```
$ mcr amqcrsta [-m Queue_Man_Name] -t DECnet
```

   Place this file in the SYS$MANAGER directory. In this example the file is named `MQRECVDECNET.COM`.

   Notes:

   a. If you have multiple queue managers you **must** make a new file and DECnet object for each queue manager.
   b. If a receiving channel does not start when the sending end starts, it is probably due to the permissions on this file being incorrect.
   c. The log file for the object is `net$server.log` in the `sys$login` directory for the application-specified user name.
   d. To enable the session control application upon every system IPL (reboot), add the preceding NCL commands to the file `SYS$MANAGER:NET$APPLICATION_LOCAL.NCL`.

Defining an LU6.2 connection

See the release notes shipped with MQSeries for Compaq OpenVMS for information about configuring SNA LU.2 connections. For instructions on how to obtain a copy of the release notes, see “Reading the release notes” on page 3.

Appendix C. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are

written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, or other countries, or both:

MQSeries	IBM
CICS	AS/400
First Failure Support Technology	BookManager
OS/390	TXSeries
IBMLink	

Lotus and Notes are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

ActiveX, Visual Basic, Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

DIGITAL, OpenVMS, Compaq, and Alpha are trademarks of the Compaq Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

R

S

T

U

V

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44–1962–842327
  - From within the U.K., use 01962–842327
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC34-5885-00