VMS Software

VSI OpenVMS

Distributed Queuing Service System Manager's Guide

Document Number: DO-DWDQSM-01A

Publication Date: April 2024

Operating System and Version: VSI OpenVMS IA-64 Version 8.4-1H1 or higher
VSI OpenVMS Alpha Version 8.4-2L1 or higher

Software Version: The Distributed Queuing Service Version 1.3

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

Distributed Queuing Service System Manager's Guide

Legal Notice

Preface

The Distributed Queuing Service (DQS) for OpenVMS Systems software enables you to print files on devices that are attached to remote DECnet systems on your network. DQS software also allows you to display the status of your jobs at any time during their processing and to delete or modify your print jobs.

The Distributed Queuing Service Version 1.3 for OpenVMS systems software product replaces the following products:

- VAX Distributed Queuing Service Version 1.2 software
- Distributed Queuing Service for OpenVMS AXP Version 1.2 software

This product is based on a client/server design, where two components of software (client and server) cooperate to make the print service function.

1. About VSI

VMS Software, Inc. (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

2. Intended Audience

This part describes how to configure and manage the DQS software on OpenVMS AXP and OpenVMS VAX systems. The System Manager’s Guide is intended for the network or system managers.

3. Related Documents

Refer to the following documents for additional information:

- *Distributed Queuing Service System User's Guide*
- *Distributed Queuing Service for OpenVMS Systems Installation Guide*

4. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: <docinfo@vmssoftware.com>. Users who have VSI OpenVMS support contracts through VSI can contact <support@vmssoftware.com> for help with this product.

5. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com.

6. Conventions

Table 1 lists the conventions used in this guide.

Table 1. Conventions

Convention	Meaning
OpenVMS system	Means both the OpenVMS AXP operating system and the OpenVMS VAX operating system.
UPPERCASE TEXT	Indicates the name of a command, a file, a parameter, a procedure, or utility.
$ PRODUCT	In interactive examples, prompts or displayed text appears in a monospace font. User input appears in a bold monospace font.
lowercase italics	Indicates variables in command syntax or examples for which the user supplies a value.
Ctrl/x	In interactive examples, a sequence such as Ctrl/x indicates that you must hold down the Ctrl key while you press another key or a pointing device button; for example, Ctrl/C or Ctrl/Z.
Return	Indicates the Return key.
[ ]	In command formats, square brackets encloses optional values. (Do not type the brackets.) In installation prompts, square brackets enclose a default value. In a file specification, square brackets serve as delimiters for a directory name.

Chapter 1. Understanding the Distributed Queuing Service

This chapter describes:

- An overview of the design of the Distributed Queuing Service for OpenVMS Systems (DQS) product
- The primary functions of the DQS software
- The functions of the DQS commands: PRINT, QSHOW, QSET/ENTRY, and QDELETE/ENTRY
- The functions of DQS software components
- The DQS notification service and how it works
- The DQS symbionts
- The use of forms with the DQS software
- The DQS server account
- Some configuration issues
- Types of errors that may occur
- How the DQS software handles security issues
- The processes created by the software
- How the DQS software supports devices
- The use of a batch job with DQS software

1.1. Overview of the DQS Software

What Is DQS Software?

The standard OpenVMS queue system enables you to use the OpenVMS PRINT command to print files on output devices that are connected to your local OpenVMS node.

The DQS software is an OpenVMS layered product that extends the standard OpenVMS queue system to a distributed system environment. The DQS software enables you to use the OpenVMS PRINT command to print files on output devices that are connected to remote OpenVMS nodes in your network. You can also delete, and obtain or change the status of your DQS print jobs.

Why Use DQS Software?

Installing and configuring the DQS software in your network can provide:

- Printing services to workstation users (and others) who do not have attached printers
- The sharing of expensive or unique printing devices among the users in your distributed system
- The ability for your users to distribute reports efficiently to remote sites within your company or organization, because DQS software works in both wide area and local area networks.

Understanding DQS Clients and Servers

The DQS software is based on a client/server design, which consists of two cooperating software components (client and server).

The following terms are associated with DQS servers:

Server nodes: Nodes in your network that have connections to output devices (for example, printers) that you want to make available to the users on other nodes (client nodes) in your network.

Remote queue: DQS queue on the server node that directs print jobs from client nodes to the connected printer. (Standard OpenVMS print queues are queues that direct print jobs to printers connected to the local node.)

Server software: DQS software that you must install and configure on server nodes.

The following terms are associated with DQS clients:

Client nodes: Nodes whose users are able to print files on printing devices that are connected to remote DQS server nodes.

Client queue: DQS queue on the client node that directs print jobs to the associated remote queue on the DQS server node.

Client software: DQS software that you must install and configure on client nodes.

A node can be only a client, or both a client and a server. Each client can be configured to send print jobs to any server. Similarly, each server can be configured to accept print jobs from any client. Chapter 2 and Chapter 3 describe the procedures for configuring servers and clients, respectively.

Using DQS Software in a Cluster

In a VMScluster environment, a server is potentially every node in a cluster. Similarly, a client is potentially every node in a cluster. Therefore, if a user logs on to different cluster members at different times, the DQS software is always available.

To use the DQS software in a cluster environment:

1. Install the DQS software.
2. License each node in the cluster to be a DQS node.
3. Invoke DQS$STARTUP.COM from each cluster member node.

While DQS software can be used on selected nodes of a cluster, doing so does not take advantage of the flexibility of a cluster.

Network Requirements for DQS Software

DQS software uses the DECnet network to communicate between clients and servers and to perform its functions. The DQS software requires:

- The client and server to communicate for the print request to be processed by the server.

  However, users can queue print jobs at any time, even if the client system is not in communication with the desired server system. Similarly, a server can continue to process print jobs, even if the client from which the job was received is not in communication with the server.

- A connection between a client and a server remains long enough to transfer the complete file (or files) to be printed.

  If the connection is broken before the entire print job is transferred to the server, the DQS software attempts to transfer the entire print job after the connection is reestablished.

- Communication must exists between the client and the desired server to examine a remote queue or to delete or modify a job in a remote queue.

- A connection must be established from the server to the user's client system for a user to receive job notification messages (described in the section titled The DQS Notification Service).

Phase IVand Phase V Networks

The DQS software can be used in both DECnet Phase IV and DECnet/OSI Phase V networks. DQS software also operates in properly configured networks that use both networking products. DECnet/OSI systems can be set up to use synonyms to communicate with Phase IV systems.

DECnet Phase IV networks limit node names to six characters. DECnet/OSI Phase V networks allow the use of an expanded node name (full names). DQS clients and servers can handle node names with up to 150 characters with no embedded blank spaces.

1.2. Primary Functions of the DQS Software

Overview

The major functions of the DQS software are initiated by the PRINT, QSHOW, QSET/ENTRY, and QDELETE/ENTRY commands. DQS software uses:

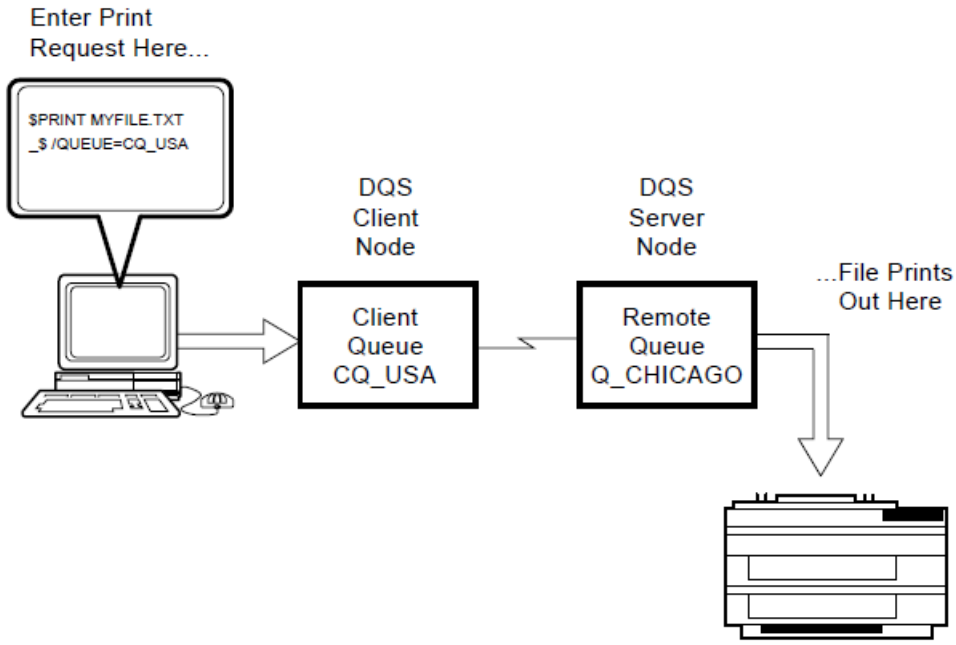
- The standard OpenVMS PRINT command to initiate DQS print requests

- Modified versions of the OpenVMS SHOW, SET, and DELETE commands to show, modify, and delete DQS print requests

Primary Functions

Figure 1.1 shows the primary functions of the DQS software.

Figure 1.1. Overview: Primary Function of the DQS Software



The Process

The stages shown in Figure 1.1 are summarized as follows:

Stage	Description
1	A user enters a PRINT command on a client node specifying a DQS client queue.
2	DQS software copies the file to be printed from the client queue to the server node, and enters the print job in the remote queue on the server.
3	The remote queue sends the file to the server's output device and the file is printed.

Summary of DQS Commands

Most of the standard OpenVMS command qualifiers are fully supported for the DQS versions of the PRINT, SHOW, SET, and DELETE commands. Table 1.1 summarizes the DQS commands.

Table 1.1. Summary of DQS Commands

Stage	Description
PRINT	The OpenVMS PRINT command places a print job in a client queue on a client system. The DQS symbiont transfers the print request to the associated server system, where it is placed in the associated remote queue and directed to an output device for printing.
QSHOW	Displays the status of a DQS print job that is in either a client or a remote queue.
QDELETE/ ENTRY	Prevents a print job from being processed or terminates its processing.
QSET/ENTRY	Modifies the parameters of a print job after the job has been queued but before it has been processed.

Refer to *Distributed Queuing Service for OpenVMS Systems User's Guide* for more information on the DQS commands and their qualifiers.

1.3. Functions of the DQS Commands

PRINT Command Functions

Figure 1.2 shows the functions of the PRINT command using a DQS queue.

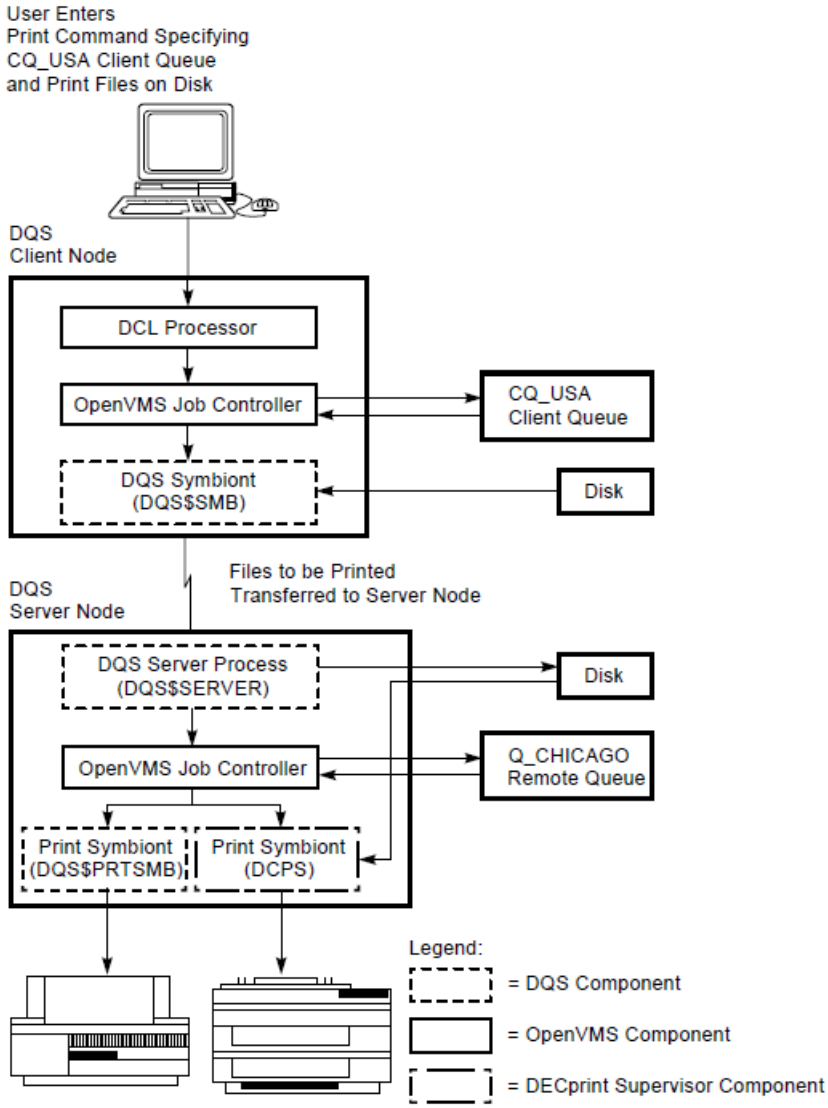
The Process

The stages in the PRINT process are summarized as follows:

Stage	Description
1	A user issues a PRINT command specifying the DQS client queue CQ_USA.
2	The standard OpenVMS DCL command language intepreter interprets the PRINT command.
3	The OpenVMS job controller in the client queue on the client node queues the print job to the DQS client queue.
4	The client symbiont, DQS$SMB, takes the print job from the client queue and transfers the job to the DQS$SERVER process on the server node.
5	The DQS$SERVER process receives the print job from the client symbiont, writes the file to a disk on the server node, and places the print job in the remote queue Q_CHICAGO.
6	The print symbiont on the server retrieves the file when the file is ready to be printed and directs the file to the output device for printing.

After the printer prints the file, the DQS$NOTIFIER process on a server deletes the file from the DQS server account (see the section titled The DQS Server Account). If a user requests notification of job com pletion, the DQS$NOTIFIER attempts to notify the user before it deletes the files associated with the print job.

Figure 1.2. The DQS PRINT Command Process



Other DQS Commands

Figure 1.3 shows the functions of the QSHOW, QSET/ENTRY, and QDELETE/ENTRY commands.

The Process

The stages in the DQS QSHOW, QSET/ENTRY, and QDELETE/ENTRY command processes are summarized as follows.

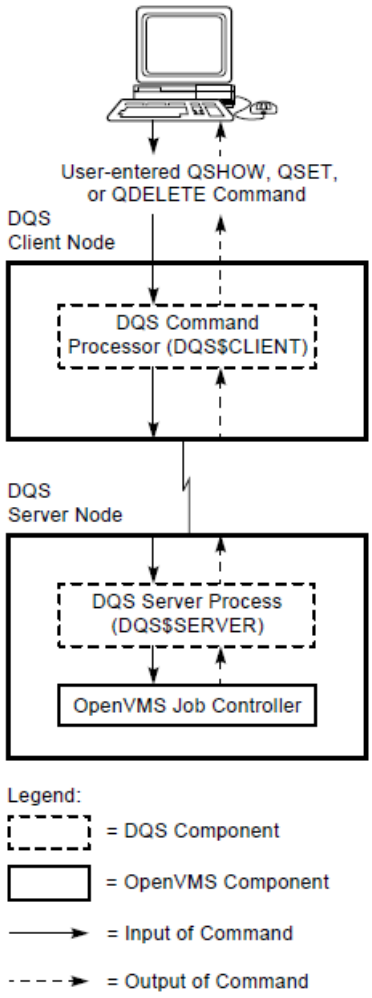
Stage	Description
1	The DQS comman d interpreter DQS$CLIENT interprets the command and transfers the request to the
2	The DQS$SERVER process receives the command request from the client node and passes the request to the
3	The OpenVMS job controller performs the command request and returns the appropriate information to the DQS$SERVER process.

Stage	Description
4	DQS$SERVER then transfers this information to the DQS$CLIENT process on the client node.
5	DQS$CLIENT format s the information and directs the command output to the user who initiated the command.

Commands with OpenVMS Queues

The DQS QSHOW, QSET/ENTRY, and QDELETE/ENTRY commands also work on standard OpenVMS print queues. For an entry in a standard OpenVMS queue, QSHOW, QSET/ENTRY, and QDELETE/ENTRY do not send information to a server.

Figure 1.3. DQS QSHOW, QSET/ENTRY, and QDELETE/ENTRY Commands Processes



1.4. Functions of DQS Software Components

Summary of Components

Table 1.2 summarizes the major software components of the DQS software and describes their functions.

Table 1.2. Functions of the DQS Software Components

Component	Name	Description
DQS command interpreter	DQS$CLIENT	The DQS command interpreter is located on clients and interprets the DQS QSHOW, QSET/ENTRY, and QDELETE /ENTRY commands. The standard OpenVMS command language interpreter interprets the PRINT command. The standard OpenVMS batch/print system enters the print job into a client queue.
DQS symbiont	DQS$SMB	The DQS symbiont runs on clients. Each client queue is serviced by an OpenVMS process that runs the DQS$SMB image. The DQS symbiont process takes print jobs from a client queue and transfers the jobs to a process on the associated server.
DQS server program	DQS$SERVER	The DQS server program is located on servers and receives print jobs from the DQS symbiont DQS$SMB on clients. The server program writes the file(s) to be printed to the server disk and places the print request in the associated remote queue. DQS$SERVER also processes QSHOW, QSET/ENTRY, and QDELETE/ENTRY comman d requests received from clients. The DQS server process is also located on clients to receive job completion messages from the DQS$NOTIFIER process on servers. DQS$SERVER on a client broadcasts job completion messages sent by the DQS$NOTIFIER to each user who owns a particular print job that has completed.
Notification program	DQS $NOTIFIER	The DQS notification program is located on servers and periodically scans DQS print jobs in the remote queues to determine which print jobs have completed. After a print job completes, DQS$NOTIFIER informs the DQS$SERVER process on the client from which the print job originated if /NOTIFY is specified.
Modified OpenVMS print symbiont	DQS$PRTSMB	The DQS print symbiont, DQS$PRTSMB, is located on servers and is a modified version of the standard OpenVMS print symbiont, PRTSMB. The modified DQS symbiont should replace the standard OpenVMS print symbiont so that client-specific information is printed on the flag, burst, and trailer pages of the print output. See The DQS Symbionts for more information on the DQS symbiont.

1.5. The DQS Notification Service

Sending Messages to Users

If you specify /NOTIFY qualifier with the PRINT command, the notification service sends job completion messages when print jobs have completed.

DQS software returns two notification messages:

- The first when the user’s print job has transferred to the server.

- The second after the job has printed on the server.

Deleting Jobs from Remote Queue

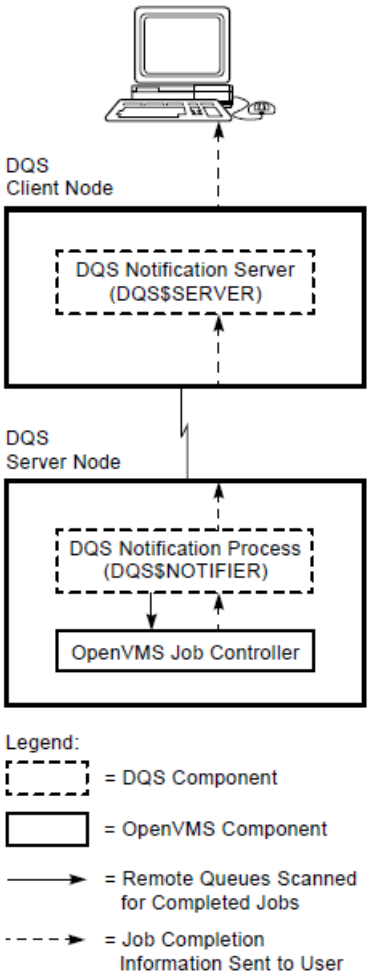
The notification process on the server node, DQS$NOTIFIER, also has the task of deleting completed print jobs from the remote queue. The DQS$SERVER program submits all DQS print jobs with the /RETAIN=ALWAYS qualifier. DQS print jobs remain in DQS queues after the job is printed until the DQS$NOTIFIER deletes them.

If a user requests notification of job com pletion, DQS$NOTIFIER attempts to notify the user before it deletes the files associated with the user's print job.

How It Works

Figure 1.4 shows how the DQS notification service works.

Figure 1.4. Function of the DQS Notification Service



The Notification Process

The DQS notification process shown in Figure 1.4 works like this:

Stage	Description
1	The DQS notification process on the server, DQS$NOTIFIER, periodically scans DQS jobs in the remote queues to determine which print jobs have completed.

Stage	Description
2	When a print job com pletes, the DQS$NOTIFIER checks if the user specified /NOTIFY and sends a completion message to the DQS$SERVER process on the client node.
3	The DQS$SERVER on the client receives the job completion message and a message is sent to the user who owns the job that has completed.

For more information on using the /NOTIFY qualifier with the PRINT command, refer to the "Printing Files Using DQS Software" chapter in *DEC Distributed Queuing Service for OpenVMS Systems User's Guide* and the *OpenVMS DCL Dictionary*.

1.6. The DQS Symbionts

Overview

The DQS software contains two symbionts DQS$SMB and DQS$PRTSMB, which are described in this section.

DQS$SMB

DQS$SMB runs on client systems. The symbiont takes DQS print jobs from client queues and transfers them to the associated server node.

DQS$PRTSMB

DQS software provides its own print symbiont, DQS$PRTSMB, which is a modified version of the standard OpenVMS print symbiont, PRTSMB. (The DQS$PRTSMB symbiont is upwardly compatible with the OpenVMS PRTSMB symbiont.) DQS$PRTSMB causes the client job number, the client node name, and the time a print job was queued on the client to be printed on the flag, burst, and trailer pages of DQS print output. With the DQS print symbiont controlling each remote queue on a server, the users of that server see the same job information on their print output that they saw displayed when they entered the print request on the client node.

If the standard OpenVMS print symbiont does not handle banner pages to your satifiaction, replace it with the DQS print symbiont, DQS$PRTSMB. If DQS$PRTSMB does not control a remote queue on a server, the standard OpenVMS print symbiont prints the job number of the print job in the remote queue, as well as the time that the print job was queued on the server on the DQS print output.

DQS$PRTSMB can only replace the OpenVMS print symbiont.

For a description of the procedures for enabling the DQS print symbiont to control remote queues, see Chapter 2.

Refer to Appendix B for information on how DQS$PRTSMB relates to other symbionts.

1.7. Defining Forms on Client and Server Nodes

Overview

You must define all forms that you intend to specify on both the client and its associated servers. To synchronize form definitions on your clients and servers, you can run the DQS$CLIENT program on the client system (see Chapter 3).

No Form on Server

If you attempt to print using a form that is not defined on the associated server, the client queue accepts the job and the system prompt returns ($). However, your print job remains in the client queue with the error status %DQS-F-NOSUCHFORM. A QSHOW command specifying the client queue where you queued the job shows the retained job and the error message.

Different Values

Because DQS software transfers only a form's name from a client to its server, a form may have different values on different DQS systems. When the same form is defined differently throughout your network, the server's form definition is used to print a job.

VSI strongly recommends that you give forms with the same name the same definitions throughout your network. However, you may want to have different form definitions because it enables users on client systems to maintain their local form definitions to use when printing locally.

Chapter 3 describes the procedures for synchronizing form definitions.

1.8. The DQS Server Account

Overview

The DQS installation procedure creates a DQS account, DQS$SERVER, and its associated directory [DQS$SERVER] on all DQS client and server nodes.

On a Server

A DQS server process is run from the DQS$SERVER account when a connection request is received from a client node. The directory associated with this account [DQS$SERVER] stores files received from client nodes until the files are printed on the server node's output device. After a print job completes, the files associated with the job are deleted from the server account directory.

On a Client

The DQS$SERVER account is located on clients to receive job com pletion messages from the DQS $NOTIFIER process on the server. DQS$SERVER on a client broadcasts job com pletion messages to each user who owns a completed print job, if the user specified /NOTIFY on the PRINT command.

1.9. The Significance of Parameter 8 in DQS Printing

Overview

The OpenVMS PRINT command allows you to pass from one to eight optional parameters with a print job. DQS software uses parameter 8 to store print job information. Place your print parameter values in parameter 1 to parameter 7.

Use in DQS Printing

DQS software uses the information in parameter 8 to:

- Validate a user's request
- Delete a print job
- Modify a print job

The information stored in parameter 8 includes:

- Client job number
- Job owner; on OpenVMS systems this usually consists of:
  - Client node name
  - User's name on the client
- Time the job was queued on the client

Error Messages

DQS software issues an error (%DQS-E-P8RE SERVED) if you attempt to specify parameter 8 with either a PRINT or a QSET/ENTRY command.

For more information about:

- The %DQS-E-P8RESERVED error message, see Appendix A
- The format of the information stored in parameter 8, see Appendix B
- The use of the /PARAMETERS qualifier, see the "Printing Files Using DQS Software" chapter in *Distributed Queuing Service for OpenVMS Systems User's Guide*.

1.10. Configuring DQS Software

Client and Server Installation

When you run the DQS installation procedure, you are asked whether you want to install DQS server software. You are given this option because not all DQS nodes in your network require server software (for example, those nodes without connections to output devices). However, most or all nodes in

your network require client software so that their users can have access to remote printing resources. Therefore, client software is always installed when you install DQS software.

Client and Server Configuration

A client must be configured to work with its corresponding servers (and vice versa) in order for DQS software to function. You can configure a server to serve as many clients as you want.

Similarly, you can configure a client to use the services of any number of servers. There are no restrictions on the number of DQS servers or clients that you can have in your network.

Restrictions to DQS Configurations

The DQS software is suitable for all OpenVMS nodes in DECnet networks. DQS software can be used in small networks consisting of two nodes and in large networks consisting of many nodes. DQS software operates in either local area networks or wide area networks.

DQS software is limited only by the ability of DQS servers to accept and maintain a large number of simultaneous network connections (that is, logical links) to DQS clients. The maximum number of logical links that a server can support is a parameter that you can define using the Network Control Program (NCP) on DECnet Phase IV systems and the Network Control Language (NCL) on DECnet/OSI Phase V systems. The value of this parameter depends upon many factors, such as the server's processor speed and its typical work load.

Each logical link from a client creates one process on a server. A server must be configured to allow for enough processes. (See the section entitled Processes Created by DQS Software, which describes the processes created by the DQS software on both clients and servers.)

In addition to handling its other network applications, a server should be able to accept simultaneous connections from all clients that it serves. However, all clients trying to access the server at the same time is unlikely. Any client that attempts to print on the server after the maximum number of logical links is reached, automatically attempts to reconnect to the server after a short wait. The QSHOW, QSET/ENTRY, and QDELETE/ENTRY command functions do not attempt this automatic reconnection.

1.11. Using Daisy-Chained Queues

Purpose

You can configure the DQS software in your network so that a client queue corresponds to a remote queue, which in turn, corresponds to another remote queue on another server. This configuration is called daisy-chained queues. You can daisy-chain queues across as many servers as you want.

Benefits

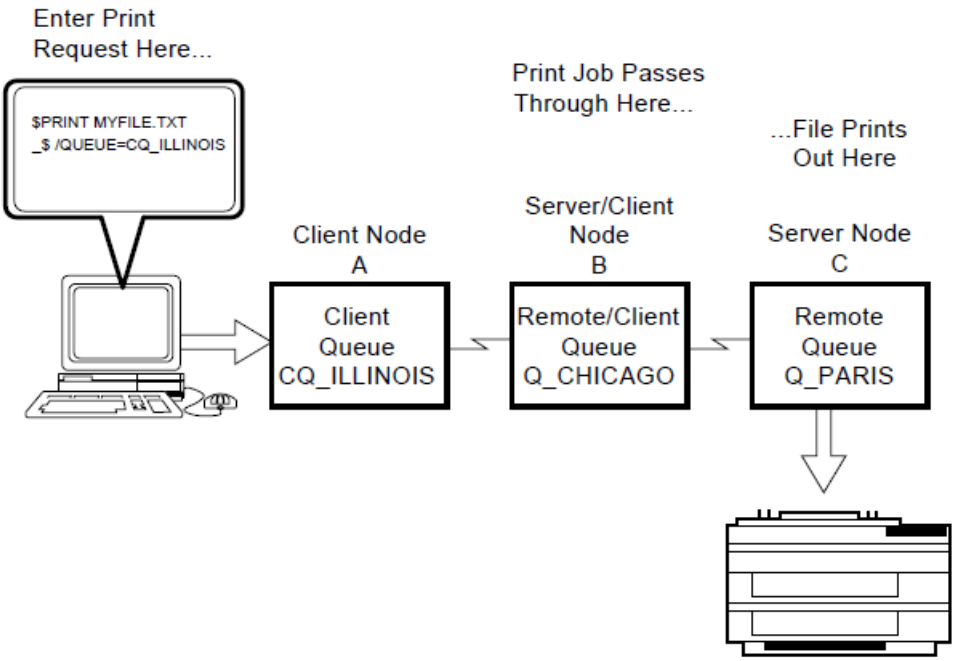
Daisy-chained queues are useful when a printer on a server becomes temporarily unavailable. Users who want to send jobs to the disabled server can still do so if the server's system manager sets up the remote queue to redirect their print output to another server's output device. No change is made to the user's client system, and the users may not even know that the printer was unavailable.

Daisy-chained queues are also useful when a network link from a client to its server is only occasionally available. With daisy-chained queues you can effect a "store and forward" function, where a job is transferred to a reachable, intermediate server that stores the job until the desired server is reachable.

Example

Figure 1.5 depicts a daisy-chained queue configuration.

Figure 1.5. Daisy-Chained Queues



Process

The daisy-chain process shown in Figure 1.5 works as follows:

Stage	Description
1	A user enters a print job in the client queue CQ_ILLINOIS on client node A.
2	DQS software transfers the job to node B and enters it in the remote queue Q_CHICAGO.
3	DQS software then transfers the job from node B to node C, and enters it in the remote queue Q_PARIS on node C.
4	The job is printed on node C's printer.

Avoiding Loops

Avoid creating a loop when configuring daisy-chained queues. If you configure a daisy-chained queue to direct a job back to the system where the job was originally queued, the job terminates on that system with the error, %DQS-E-LOOP.

For More Information

Refer to *Distributed Queuing Service for OpenVMS Systems Installation Guide* for more information on the installation of DQS software.

Chapter 2 and Chapter 3, respectively, describe the procedures for configuring servers and clients after installation of the software.

Chapter 2 describes the procedures for setting up daisy-chained queues.

1.12. DQS Error Conditions

Error Detection

When DQS software detects an error condition, the error is sent to the OpenVMS operator communication process (OPCOM). If an error is detected in a server process, it is also written to a NETSERVER.LOG log file on the server.

Types of Errors

There are three basic types of errors that can occur in DQS printing.

- A print job fails because a network link to the server is not available or because there is insufficient disk space on the server.

  The DQS symbiont handles this type of error by automatically trying to reconnect to the server node after a five-minute wait (this value is not changeable).

- A print job cannot transfer to a server.

  The DQS symbiont handles this type of error by keeping the print job in the client queue with the error status "Retained on error." A QSHOW command specifying the client queue in question displays the specific reason for the error. In this case, you must requeue, release, or delete the print job.

  This type of error is depicted in Example 1.1.

- A client queue is stopped by the DQS symbiont.

  This type of error occurs when a user attempts to print to an invalid remote queue or to a server node whose name is unknown to the client node. A QSHOW command specifying the client queue in question displays the specific reason for the error.

  This type of error is depicted in Example 1.2.

Examples

Example 1.1 shows an error detected by the client symbiont, DQS$SMB. The error does not affect the operation of the symbiont but causes the job responsible for the error to terminate with error status. The affected job remains in the client queue with the error status "Retained on error." In this case, you must either requeue or delete the print job. In Example 1.1, the client node has a form defined that is not defined on the server. This prevents the print job from transferring. A QSHOW command specifying the client queue where the job was queued shows the retained job and the specific reason for the error.

Example 1.1. A DQS Error That Prevents a Print Job From Transferring

```
$ QSHOW QUEUE1 Return
** Remote queue - [ QUEUE1 => SYS$PRINT, on NODEB:: ]
Terminal queue SYS$PRINT, on NODEB::
============================to be
 transferred============================
```

```
Server queue QUEUE1, on NODEA::, mounted form DEFAULT
Jobname Username Entry Blocks Status
------- -------- ----- ------ ------
JOB PAUL 925 1 Retained on error
%JBC-E-NOSUCHFORM, no such form
```

Example 1.2 shows an error detected by the client symbiont, DQS$SMB, that caused the client queue to stop. The DQS$STATUS_*queue-name* logical name is defined to be the error status. This action makes it easy to see the reason for the error by issuing a QSHOW command for the client queue. In

Example 1.2, the server node name is not known on the client. Therefore, the symbiont cannot process any jobs for that queue and stops the queue.

Note

Remember to deassign the DQS$STATUS_*queue-name* logical name when you restart the queue.

```
$ DEASSIGN/SYSTEM/USER DQS$STATUS_queue-name Return
```

Example 1.2. A DQS Error That Stops a Client Queue

```
$ QSHOW QUEUE1 Return
***** Remote queue NODEB::QUEUE1 not accessible *****
%SYSTEM-F-NOSUCHNODE, remote node is unknown
==============================to be
 transferred==============================
Server queue QUEUE1, stopped, on NODEA::, mounted form DEFAULT
       remote system error: remote node is unknown
   Jobname     Username    Entry    Blocks    Status
   -------     --------    -----    ------    ------
   LOGIN        YOUNG        571         2    Pending
```

Refer to Chapter 4 for guidelines on how to troubleshoot error conditions that involve DQS software.

1.13. DQS Security Issues and Access Rights

Restricting Access

By default, all client nodes have access to a server.

You can restrict access to a server node only by client system, not by individual client users.

Chapter 2 describes the procedures for granting or denying client access to a server.

1.14. Processes Created by DQS Software

On a Client

On a client node, every client queue that you create (that is not a generic or logical queue) results in the creation of one process.

Also, while a user is being notified of print job completion, a process is temporarily created.

On a Server

For server nodes, every incoming connection from a client node results in the creation of one process. Also, the notification process on a server uses one permanent process.

1.15. Devices Supported by DQS Software

Supported Devices

DQS software does not run devices; rather, it places print jobs in queues for processing. DQS software supports devices by placing jobs in queues, if the device

- Is driven by an OpenVMS print symbiont
- Does not use parameter 8 in job requests

DQS software does not support symbionts that require job information outside of that stored or maintained by the OpenVMS queue system.

1.16. Batch Jobs and DQS Printing

DQS software supports the queuing of print jobs for processing on remote nodes in your network. It does not support the queuing of batch jobs.

Do not configure batch queues to be DQS printer queues. DQS software does not prevent you from configuring batch queues, but the security of the system decreases if you do so.

Chapter 2. Configuring and Managing a DQS Server

This chapter presents an overview of the management tasks required by a DQS server and describes how to do the following:

- Modify definitions of special logical names associated with the DQS server
- Set up DQS remote queues
- Control DQS client access to servers and their remote queues
- Determine how DQS clients are accessing a server
- Control the note on the banner page of a print job
- Enable DQS status messages for queues
- Enable the DQS print symbiont to control remote queues
- Specify the scanning interval for the DQS server notification process
- Specify a directory for Print Server log files
- Specify the priority of the server process
- Specify the maximum priority of DQS print jobs
- Specify the duration of links to client nodes
- Move the server account directory to a new device
- Configure server software for a VMScluster environment
- Configure daisy-chain queues

The last section contains a summary of the server management tasks.

2.1. Overview of Server Management

Main Tasks

Configuring and managing a DQS server primarily involves:

- Defining server system logical names in the SYS$MANAGER:DQS$SYSTARTUP.COM file

  All site-specific definitions that affect the behavior of the software belong in this file. If this file does not exist on your system, copy the DQS$SYSTARTUP.TEMPLATE to DQS$SYSTARTUP.COM and edit it to include the desired server logical name definitions.

- Setting up access to remote queues in the SYS$MANAGER:DQS$SERVER_CONFIG.TXT file

The SYS$STARTUP:DQS$STARTUP.COM procedure

- Invokes the SYS$MANAGER:DQS$SYSTARTUP.COM procedure to make changes permanent.

- Processes the SYS$MANAGER:DQS$SERVER_CONFIG.TXT file to set up DQS client node access to remote queues on DQS server systems.

Do not edit SYS$STARTUP:DQS$STARTUP.COM.

Using SYSMAN

When directed to restart DQS software in a VMScluster, you can use one of the following methods:

- Invoke SYS$STARTUP:DQS$STARTUP.COM on each node in the cluster.
- Execute the SYSMAN utility.

2.2. Defining Server System Logical Names

Procedure

To make permanent changes to your server configuration in the DQS$SYSTARTUP.COM file, do the following:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.

2	Add or change the definitions of the desired server logical names. Make sure that you define the logical name systemwide and in executive mode:
	
	`$ DEFINE/SYSTEM/EXECUTIVE_MODE`
	`_$ logical-name values Return`
	
	For example, to set the server process to run at a priority of 5, enter:
	
	`$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$PRIORITY 5 Return`

3	Exit from the file SYS$MANAGER:DQS$SYSTARTUP.COM.
4	If the change is required immediately, do one of the following: • Also enter the command at the DCL prompt. • Execute the DQS startup command procedure. `$ @SYS$STARTUP:DQS$STARTUP Return` Otherwise, the change takes effect the next time the SYS$STARTUP:DQS $STARTUP.COM is executed or the node is rebooted.

Adding the logical name definitions to the SYS$MANAGER:DQS$SYSTARTUP.COM file ensures that the logical name is defined each time your server reboots or each time the SYS$STARTUP:DQS $STARTUP.COM procedure is executed on a cluster member.

Precaution

Do not execute the DQS$STARTUP.COM procedure unless it is necessary. Executing DQS $STARTUP.COM terminates any DQS network operations in progress. This may cause files transferred from a client to be left on the server with no association to the client job. When this happens, the server

directory clean-up process, DQS$CLEANSRV.COM, detects and deletes the stray files left on the server disk. (DQS$STARTUP.COM submits the DQS$CLEANSRV.COM procedure at start-up time.)

Changes at DCL Prompt

You can also define some DQS server logical names by entering the definition at the DCL prompt. These changes to server configuration:

- Take effect the next time the client establishes a connection to the server.
- Remain in effect until the server is rebooted or the DQS$STARTUP.COM is executed again.

2.3. Modifying Definitions of Certain Logical Names

Logical Names

For changes to the following logical name definitions to take effect, you must make your changes in the site-specific startup file DQS$SYSTARTUP.COM and restart all server processes:

- DQS$LOG_ACCE SS
- DQS$ACCOUNTING_BY_SYSTEM
- DQS$NOTIFY_CYCLE_TIME
- DQS$PRIORITY
- DQS$MAX_PRIORITY
- DQS$IDLE_TIME

Restarting Server Processes

To restart the server processes, execute the DQS$STARTUP.COM command procedure.

```
$ @SYS$STARTUP:DQS$STARTUP Return
```

Executing DQS$STARTUP.COM terminates any DQS network operations in progress.

2.4. Before You Set Up Remote Queues

Ensure Queues Exist

A queue must exist before you can set up the queue as a DQS remote queue. You can use existing queues or create (or initialize) new queues. (Refer to the *OpenVMS System Manager's Manual* for information on how to initialize a queue.)

For New Installations

If you are installing this product for the first time, the DQS installation provides the following server configuration file to which you need to add your queue configurations:

```
SYS$MANAGER:DQS$SERVER_CONFIG.TXT
```

For Product Upgrades

If you are upgrading from a version previous to Version 1.3, the installation procedure:

- Creates the SYS$MANAGER:DQS$SERVER_CONFIG.TXT file
- Attempts to translate your current queue definitions to the format of this file
- Provides you with messages in the SYS$UPDATE:DQS$CONVERT_SECURITY.LOG file
- Saves your queue definitions from the previous version by renaming the old startup file to DQS $STARTUP.COM_OLD

Compare the queue definitions in the DQS$SERVER_CONFIG.TXT file to those in DQS $STARTUP.COM_OLD. You may need to edit DQS$SERVER_CONFIG.TXT to ensure that your queue definitions are as you want them.

2.5. Understanding How to Set Up Remote Queues

Keywords

You enable network access to queues and control authorization to the server node and remote queues by using the following keywords in the DQS$SERVER_CONFIG.TXT file:

QUEUE: Declares a queue as a DQS network accessible queue. You must specify this keyword to allow network printing.

You also use the QUEUE keyword to specify which authorized client nodes have access to the queue.

DENY_NODE: Denies a client node authorization to the DQS servers on the server node. This keyword takes precedence over all other keywords.

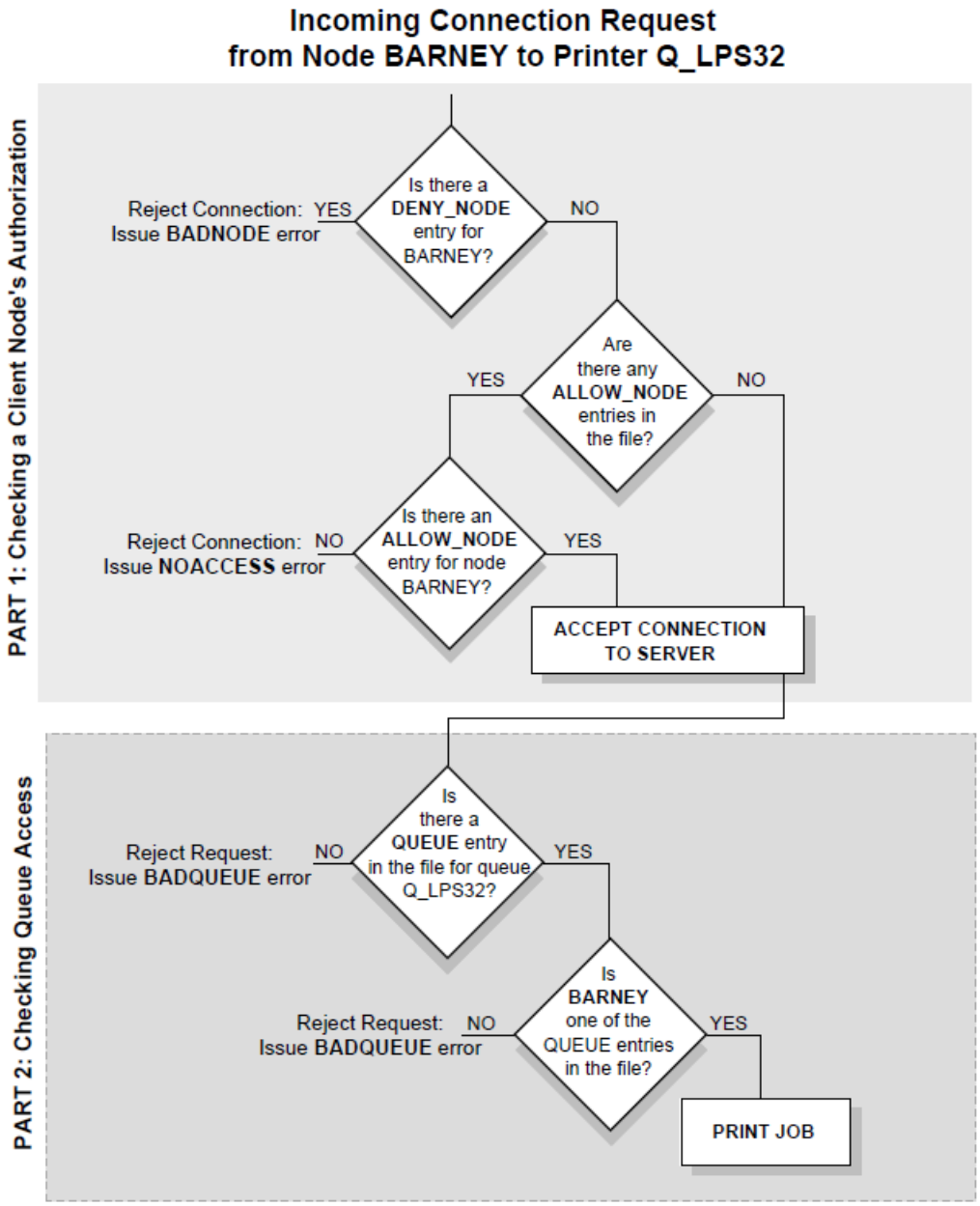
ALLOW_NODE: Grants only the client nodes you specify authorization to the DQS servers on the server node. All other nodes are denied authorization.

The QUEUE keyword differs from the other keywords. QUEUE controls access to **queues**, whereas ALLOW_NODE and DENY_ NODE authorize access to the **DQS servers** on the server node.

Client Authorization Process

Figure 2.1 shows how the configuration file checks that the client node is authorized to access the server node and remote queue.

Figure 2.1. Checking a Client Node's Authorization



2.6. Understanding the Configuration File

Features of the File

When you edit the DQS$SERVER_CONFIG.TXT file, notice the following:

- The file is a simple text file.
- Entries are not case-sensitive.
- Entries are specific keywords and their arguments.
- Valid keywords and their arguments are as follows:

```
Keyword     Argument 1            Argument 2
-------     ------------------    ------------------
QUEUE       queue_name            [client_node_name]
ALLOW_NODE  client_node_name
DENY_NODE   [client_node_name]
```

You can the use wildcard character (*) as an argument shown in brackets ([ ]).

- Keywords are the first non-whitespace character on a line.
- Keywords and arguments are separated by at least one space.
- Arguments are either simple names or complex names.
  - Simple names contain no spaces and do not span multiple lines. Simple names can contain any printable character, except the space character, the exclamation point (!), and the double quotation mark (").
  - Complex names have double quotation marks (") as delimiters at the beginning and the end of the name. Complex names can contain any printable character (except the double quotation mark (")) and can span multiple lines in the file.
- The exclamation point (!) marks a comment. The DQS software treats lines beginning with an exclamation point as comments and ignores them. Likewise, an exclamation point on a line that is not part of a complex name causes the rest of the line to be treated as a comment and ignored.
- The asterisk character (*) is the wildcard character. An argument that consists of only the wildcard character implies that all client nodes are affected by the operation. For example:

```
DENY_NODE *         ! means deny access to ALL nodes
QUEUE Q_PARIS *     ! means allow ALL nodes access to Q_PARIS
```

2.7. Setting Up Remote Queues

Setting up remote queues involves editing DQS$SERVER_CONFIG.TXT and:

1. Enabling network access to the queues with the QUEUE keyword. (This step is required.)
2. Optionally denying specific client nodes the authorization to access the DQS servers on the server node.
3. Optionally granting only specified client nodes the authorization to access the DQS servers on the server node.
4. Making the changes take effect.

The following sections explain how to do these tasks.

2.8. Enabling Network Access

Procedure

Enable network access to the queues with the QUEUE keyword. (This step is required.)

Specify one QUEUE entry for each remote queue on your system. This allows the remote queue to be DQS network accessible to authorized client nodes.

By default, all client nodes in your network are authorized to access the DQS servers on your node.

Unlimited Queue Access

You can use the asterisk (*) as a wildcard to allow all authorized client nodes access to the print queues.

Example

For example, the following entry allows all authorized client nodes access to the print queue Q_CHICAGO:

```
QUEUE Q_CHICAGO *
```

Limited Queue Access

To allow only certain authorized client nodes access to your queue, enter a QUEUE entry for each node that you want to allow access to that particular queue.

Example

For example, the following entries allows authorized client nodes APPLE, BANANA, and CHERRY access to Q_PARIS:

```
QUEUE Q_PARIS APPLE      ! Allow authorized client nodes APPLE,
QUEUE Q_PARIS BANANA     ! BANANA, and CHERRY access to
QUEUE Q_PARIS CHERRY     ! my queue Q_PARIS
```

Where to Go from Here

Enabling network queue access is the only required step. If you want to grant or deny certain client nodes authorization to DQS servers on your server node, continue to the next sections. Otherwise, see the section titled "Making Your Changes Take Effect."

2.9. Denying Client Node Authorization to the DQS Server

Purpose

Denying client nodes authorization to the DQS servers is primarily intended to shut out clients that inappropriately and repeatedly establish connections to a server. Denied authorization takes precedence over any other form of client access control.

Procedure

To deny a client node authorization to a DQS server on the server node, specify the name of the client node as the first argument to the DENY_NODE keyword.

Example

For example, the following entry denies client node APPLE from accessing DQS servers on the server node:

```
DENY_NODE APPLE
```

Denying All Clients

To deny all client nodes authorization to the DQS servers, add an entry with the asterisk (*) as the first argument to the keyword:

```
DENY_NODE *          ! Deny authorization to all nodes
```

This restricts all client nodes access to your DQS servers.

2.10. Granting Client Node Authorization to the DQS Server

Purpose

You can grant certain client nodes authorization to DQS servers on a server node. This method is useful when you want to allow a subset of client nodes access to the DQS servers on the server node.

Reminder

If you use ALLOW_NODE, keep in mind:

- You turn off the default that authorizes all client nodes access to the DQS server.
- You must specify every client node that you want to authorize access to the DQS server.
- You cannot use an asterisk (*) with the ALLOW_NODE keyword.

Procedure

Add the names of the client nodes to which you want to grant authorization to the DQS server as the argument to the ALLOW_ NODE keyword.

Example

For example, the following entries allow client nodes APPLE and CHERRY access to the DQS server:

```
ALLOW_NODE APPLE
ALLOW_NODE CHERRY
```

Using ALLOW_NODE with QUEUE

You can use the ALLOW_NODE and QUEUE keywords together as a filter in restricting access to the DQS server and queues. First you specify which clients nodes have access to the queue, then you specify which client nodes can access the DQS server.

Procedure

Use the following procedure to set up restricted access:

1. Use the QUEUE keyword to enable network access to the queues.

For example, the following entries allow all authorized client nodes access to queues Q_OHIO, Q_KANSAS, and Q_SHANNON:

```
QUEUE Q_OHIO *      ! Allow all authorized client nodes
QUEUE Q_KANSAS *    ! access to queues Q_OHIO, Q_KANSAS,
QUEUE Q_SHANNON *   ! and Q_SHANNON
```

2. Then, use ALLOW_NODE to authorize access to the DQS servers. For example, the following entries authorize client nodes APPLE and GRAPE to access the DQS server:

```
ALLOW_NODE APPLE      ! Grant authorization only to
ALLOW_NODE GRAPE      ! client nodes APPLE and GRAPE
```

Result

The combination of the ALLOW_NODE and QUEUE entries allows only a subset of client nodes (APPLE and GRAPE) access to queues Q_OHIO, Q_KANSAS, and Q_SHANNON.

Note

When using controlled access between two DQS server nodes that also act as client nodes, make sure each DQS server node is authorized access to the other node. Otherwise, notification messages may not be delivered.

2.11. Making Your Changes Take Effect

Procedure

When you are finished adding keywords to the configuration file, exit from the file and execute DQS $SERVER_UPDATE_CONFIG.COM to establish your changes:

```
$ @SYS$MANAGER:DQS$UPDATE_SERVER_CONFIG Return
```

Changes to the DQS$SERVER_CONFIG.TXT data file take effect the next time the client establishes a new connection to the server.

2.12. Determining How Clients are Accessing a Server

Methods

You can determine how and when clients access a particular server by doing any of the following:

- You can check the NETSERVER.LOG log files, which are created in the server account's directory. These files are periodically purged.
- If you enabled a server to log client access with the DQS$LOG_ACCE SS logical name, you can check the file created in the [DQS$SERVER.NODES] subdirectory of the server account to see how many times and the last time each client established a connection to the server.
- If OpenVMS accounting is enabled, you can check the process and image accounting records as described in the *OpenVMS System Manager's Utilities Reference Manual*.

With OpenVMS accounting enabled and the DQS$ACCOUNTING_BY_SYST EM logical name defined, the server maintains records of print jobs received.

Checking for Client Access

You can check the NETSERVER.LOG log files in the server account directory for records of client node access. However, these files are periodically purged. Example 2.1 shows how to view this file.

Example 2.1. Viewing NETSERVER.LOG Information

```
$ TYPE SYS$COMMON:[DQS$SERVER]NETSERVER.LOG Return
```

If you moved the directory, the log files might not be in SYS$COMMON.

Logging Client Access to a Server

DQS software counts each occurrence of a client connecting to a server if you define the DQS $LOG_ACCE SS logical name on the server. With DQS$LOG_ACCE SS defined, the server process creates a file in the [DQS$SERVER.NODES] subdirectory of the server account. This file has the name of the client. If the [DQS$SERVER.NODES] subdirectory does not already exist, the server process creates it.

Procedure

To define the DQS$LOG_ACCE SS logical:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.
2	Add the definition of the DQS$LOG_ACCE SS logical name with an arbitrary value. Its value is not important; only its presence on the server is important.
3	Save the file, SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.
4	Restart all server processes by executing the SYS$STARTUP:DQS$STARTUP.COM procedure.

Examples

Example 2.2 defines the DQS$LOG_ACCE SS logical name to be the arbitrary value ENABLED.

Example 2.2. Logging Client Access to a Server

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$LOG_ACCESS ENABLED Return
```

Example 2.3 shows a sample directory listing of the [DQS$SERVER.NODES] subdirectory that contains the names of the clients that accessed the server.

Example 2.3. Looking at DQS$SERVER.NODES

```
$ Directory SYS$COMMON:[DQS$SERVER.NODES] Return
Directory SYS$COMMON:[DQS$SERVER.NODES]
SMEDLY.;78    BUGSBY.;1407    JAZZY.;3    RAZBRY.;2
STENCL.;397   STIKNY.;24      TOKYO.;25
Total of 7 files.
```

Because there is a one-version limit for the [DQS$SERVER.NODES] subdirectory, only one file per client is created. The version number of the file corresponds to the number of logical-link connections received from the client.

Node names displayed in this file are trimmed to 8 characters. Some characters may be converted to underscores.

Enabling Client Accounting Information

With the DQS$ACCOUNTING_BY_SYST EM logical name defined on the server, the DQS software includes the client's node name as the Account field in the server's accounting record.

User node names written to this record are trimmed to 8 characters. Some characters may be converted to underscores.

Example with Logical Defined

For example:

```
$ RUN AUTHORIZE Return
UAF> SHOW MACOMBER Return
Username: MACOMBER                  Owner: Ted Macomber
Account: DEVELOPMENT                UIC: [100,1] ([MACOMBER])
. . .
UAF> EXIT Return
$ ACCOUNTING/TYPE=PRINT/FULL Return
Username:          MACOMBER       UIC:                [DQS$SERVER]
Account:           SMEDLY         Finish time:        14-FEB-1994
 11:49:29.88
Process ID:        00000D9A       Start time:         14-FEB-1994
 11:49:24.59
Owner ID:                         Elapsed time:                0
 00:00:05.29
Terminal name:                    Processor time:              0
 00:00:00.00
Remote node addr:                 Priority:           100
Remote node name:                 Privilege <31-00>:  00000000
Remote ID:                        Privilege <63-32>:  00000000
Remote full name:
Queue entry:       721            Final status code:  00040001
Queue name:        SERVER
Job name:          ACCOUNTING
Final status text: %JBC-S-NORMAL, normal successful completion
GETs from source:          180
QIOs to printer:             5
Pages printed:               5
```

Example with Logical Undefined

If you do not define this logical, the DQS client software uses the Account field specified in the client's User Authorization File (UAF) on the client for the user who issued the PRINT command. For example:

```
$ ACCOUNTING/TYPE=PRINT/FULL Return
PRINT   Job   Termination
--------------------
Username:           MACOMBER          UIC:              [DQS$SERVER]
```

```
Account:            DEVELOPMENT       Finish time:      14-FEB-1994
 11:49:07.65
Process ID:         00000D9A          Start time:       14-FEB-1994
 11:49:02.36
Owner ID:                             Elapsed time:               0
 00:00:05.29
Terminal name:                        Processor time:             0
 00:00:00.00
Remote node addr:                     Priority:          100
Remote node name:                     Privilege <31-00>: 00000000
Remote ID:                            Privilege <63-32>: 00000000
Remote full name:
Queue entry:        717               Final status code: 00040001
Queue name:         SERVER
Job name:           ACCOUNTING
Final status text:  %JBC-S-NORMAL,    normal successful completion
GETs from source:          180
QIOs to printer:             5
Pages printed:               5
```

Procedure

To define the DQS$ACCOUNTING_BY_SYST EM logical name:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.

2	Add the definition of the DQS$ACCOUNTING_BY_SYSTEM logical name with an arbitrary value.
	
	`$ DEFINE/SYSTEM/EXECUTIVE_MODE - Return`
	`_$ DQS$ACCOUNTING_BY_SYSTEM ENABLE Return`

3	Exit from the file, SYS$MANAGER:DQS$SYSTARTUP.COM.
4	Restart all server processes by executing the SYS$STARTUP:DQS$STARTUP.COM procedure.

Example

Example 2.4 defines the DQS$ACCOUNTING_BY_SYST EM logical name to be the arbitrary value ENABLED.

Example 2.4. Enabling Client Accounting Information

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$ACCOUNTING_BY_SYSTEM ENABLED Return
```

2.13. Controlling the Note on the Job Banner Page

Banner Page Note

The DQS software makes use of a print attribute to place a note on the banner page of a print job when logical names are defined in the DQS$SYSTARTUP.COM file. On the server, this logical name

definition provides the default text "Print Job processed by DQS V*n.n*". The note defined by this logical name prints on the banner page of a print job if:

- The /NOTE qualifier is not specified by the client node user on the PRINT command
- The logical name definition for the note is not specified for the client job

The note that prints on the banner page of a print job follows this order of precedence:

- Note specified by the user with PRINT/NOTE= *note*
- Note specified by a client logical name definition (DQS$CLIENT_DEFAULT_J OB_NOTE)
- Note specified by a server logical name definition (DQS$SERVER_DEFAULT_J OB_NOTE)

Options

You have the following options:

- Use the default note provided by the DQS software for the banner page
- Turn off the DQS logical definitions for the banner page note
- Define your own note for the banner page

To Turn Off the Default

To turn off the DQS default note on the server system, do the following:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.
2	Search for the following command and remove the exclamation mark (!) comment character: `$! $ DEASSIGN/SYSTEM/EXEC DQS$SERVER_DEFAULT_JOB_NOTE`
3	Save SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.

The change takes effect the next time the SYS$STARTUP:DQS$STARTUP.COM is executed.

To Change Note Text

To change the text of the note that prints on the banner page, do the following:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.

2	Search for the following command:
	
	`$!  $ DEFINE/SYSTEM/EXEC DQS$SERVER_DEFAULT_JOB_NOTE -`
	`$!  "Place your text here"`
	
	Substitute your note for the text "Place your text here" and remove the dollar sign and exclamation mark ($!) at the beginning of each line.

3	Save the file SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.

The next time the SYS$STARTUP:DQS$STARTUP.COM is executed and the note is not specified by the user or the client, your customized note prints on the job banner page.

2.14. Enabling Status Messages for Queues

Purpose

You can define the DQS$STATUS_*queue-name* logical name so that a status message is displayed each time a user enters a QSHOW comman d for the specified queue (or any DQS queue that points to that queue). This function is useful for displaying printer status messages, such as "Q_PRINTER is down until 1PM for repair."

Procedure

To define the DQS$STATUS_*queue-name* logical name:

Step	Action

1	Define the DQS$STATUS_*queue-name* logical name by entering the following at the system prompt:
	
	`$ DEFINE/SYSTEM/EXECUTIVE_MODE -`
	`_$ DQS$STATUS_queue-name "Message" Return`
	
	When defining the DQS$STATUS_*queue-name* logical make sure to enclose the message in quotation marks (""").
	
	You can define this logical name for both remote and client queues.

When you change the value of the DQS$STATUS_*queue-name* logical name, the next display of the queue shows the new message.

Example

Example 2.5 specifies that the status message "Down for repairs" displays each time a user enters a QSHOW command for the remote queue Q_PARIS.

Example 2.5. Defining a Status Message for a Queue

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$STATUS_Q_PARIS "Down for repairs" Return
```

When you want to remove the message for the queue, enter one of the following commands:

```
$ DEASSIGN/SYSTEM/EXECUTIVE_MODE DQS$STATUS_Q_PARIS Return
$ DEASSIGN/SYSTEM/USER DQS$STATUS_Q_PARIS Return
```

2.15. Enabling the DQS Print Symbiont To Control Remote Queues

Why Use the DQS Symbiont?

DQS software supplies its own print symbiont, DQS$PRTSMB, which you can use to replace the standard OpenVMS print symbiont on a server unless you use the DECprint Supervisor for OpenVMS

software. The DQS print symbiont writes client- specific job information on the burst, flag, and trailer pages of DQS print output. If you do not like the way the standard OpenVMS symbiont handles banner pages, enable the DQS print symbiont DQS$PRTSMB.

Procedure to Enable

To enable the DQS print symbiont to control a remote queue, do the following:

Step	Action
1	Stop the queue by entering STOP/QUEUE/RESET *remote-queue* at the DCL prompt.
2	Associate the DQS$PRTSMB process with the queue by entering INIT/QUEUE/PROC=DQS$PRTSMB *remote-queue* at the DCL prompt.
3	Restart the queue by entering START/QUEUE *remote-queue* at the DCL prompt.
4	VSI recommends that you also enter the SET QUEUE/RETAIN=ERROR *queue-name* command at the DCL prompt to hold jobs in the queue that complete unsuccessfully. With the /RETAIN=ERROR qualifier set, QSHOW commands display the error.

Example

Example 2.6 shows how to enable the DQS print symbiont DQS$PRTSMB to control the remote queue Q_PARIS.

Example 2.6. Enabling the DQS Print Symbiont

```
$ STOP/QUEUE/RESET Q_PARIS Return
$ INIT/QUEUE/PROC=DQS$PRTSMB Q_PARIS Return
$ START/QUEUE Q_PARIS Return
$ SET QUEUE/RETAIN=ERROR Q_PARIS Return
```

Refer to Chapter 1 for more information on the DQS$PRTSMB symbiont.

Appendix B describes how the DQS software works with other print symbionts.

2.16. Specifying the Scanning Interval for Server Notification

About the Notifier

The DQS software supplies a process on server nodes, DQS$NOTIFIER, that can send notification messages to users after their print jobs com plete. DQS$NOTIFIER runs as a detached process on a server and is started by the SYS$STARTUP:DQS$STARTUP.COM procedure at system start-up time. This process periodically scans remote queues to determine which print jobs are complete.

Procedure

You can specify the interval at which the server notification process scans the remote queue by defining the DQS$NOTIFY_CYCLE_TIME logical name to be a time in OpenVMS delta time format. By default, the notify cycle time is five minutes.

To specify the DQS$NOTIFY_CYCLE_TIME logical name:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.
2	Add the definition of the DQS$NOTIFY_CYCLE_TIME logical name with the desired time in OpenVMS delta time format.
3	Exit from the file, SYS$MANAGER:DQS$SYSTARTUP.COM.
4	Stop the DQS$NOTIFIER process.
5	Restart the DQS$NOTIFIER process by executing the SYS$STARTUP:DQS $STARTUP.COM procedure.

If you change the default time, consider the following:

- If you define the cycle time to be less than five minutes, the server's job controller and notifier process use more of the server's CPU cycles.
- If the cycle time you define is much greater than the default, users may not be notified of their job completion in a timely manner.

Example

Example 2.7 defines the interval at which DQS$NOTIFIER scans the remote queues to be every 10 minutes. The DQS$NOTIFIER process is stopped and restarted to effect this change.

Example 2.7. Specifying the Scanning Interval for DQS$NOTIFIER

Add to DQS$SYSTARTUP.COM:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$NOTIFY_CYCLE_TIME "0 00:10:00.0" Return
```

Then, enter at the DCL prompt:

```
$ @SYS$STARTUP:DQS$STARTUP.COM Return
```

2.17. Specifying a Directory for PrintServer Log Files

Overview

If your Print Server software supports the return of log files from DQS print jobs, you can specify a directory into which the log files can be placed.

You specify a directory for the Print Server log files by defining the DQS$LOG_AREA logical name to be the name of the directory.

Procedure

To specify a directory for Print Server software log files:

Step	Action
1	Define the DQS$LOG_AREA logical name to be the name of the directory by entering the following at the DCL prompt:

Step	Action
	$ **DEFINE/SYSTEM/EXECUTIVE_MODE** _$ **DQS$LOG_AREA device:[directory] Return**

If you want this logical name to be permanent, define the logical name in the SYS$MANAGER:DQS $SYSTARTUP.COM.

If this logical name is not defined, the Print Server software does not write log files for DQS print jobs.

Example

Example 2.8 specifies the directory [PRINTSERVER] on device 1DUA0: on the server as the directory for Print Server log files.

Example 2.8. Specifying a Directory for PrintServer Log Files

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$LOG_AREA $1$DUA0:[PRINTSERVER] Return
```

For more information on logging options for the Print Server Software, refer to the *Print Server Software for OpenVMS Management Guide*.

2.18. Specifying the Priority of the Server Process

Procedure

You can specify the priority at which the server process runs by defining the DQS$PRIORITY logical name to be the desired priority. Be aware that redefining the DQS$PRIORITY logical name affect s scheduling algorithms on the OpenVMS system and changes overall system performance.

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.
2	Add the definition of the DQS$PRIORITY logical name with the desired priority. If you do not define this logical name, the server process runs at priority 4 by default.
3	Save the file SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.
4	Restart all server processes by executing the SYS$STARTUP:DQS$STARTUP.COM procedure.

Example

Example 2.9 shows the command to add to the SYS$MANAGER:DQS$SYSTARTUP.COM file to set the server process to run at priority 5.

Example 2.9. Specifying the Priority of the Server Process

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$PRIORITY 5 Return
```

The next time a server process is started, the server process runs at the specified priority.

2.19. Specifying the Maximum Priority of DQS Print Jobs

Procedure

You can specify the maximum priority at which a print job can be queued or set by a user, by defining the DQS$MAX_PRIORITY logical name.

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.
2	Add the definition of the DQS$MAX_PRIORITY logical name with the desired maximum priority. If you do not define this logical name, the value of the SYSGEN parameter DEFQUEPRI is used by default.
3	Save the file SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.
4	Restart all server processes by executing the SYS$STARTUP:DQS$STARTUP.COM procedure.

Example

Example 2.10 specifies that print jobs can be queued at no higher than priority 42.

Example 2.10. Specifying the Maximum Priority for Queuing Print Jobs

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$MAX_PRIORITY 42 Return
```

2.20. Specifying the Duration of Links to Client Nodes

Overview

You can specify the duration that a server maintains logical links to inactive clients. A client is considered inactive if it is not sending print jobs to a server.

Procedure

You specify the duration of logical links by defining the DQS$IDLE_TIME logical name to be the time duration in minutes in OpenVMS delta time format.

Step	Action
1	Edit the file, SYS$MANAGER:DQS$SYSTARTUP.COM.
2	Add the definition of the DQS$IDLE_TIME logical name with the desired duration time in minutes in OpenVMS delta time format. If you do not define this logical name, an idle time of 15 minutes is used by default.
3	Save the file SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.

Step	Action
4	Restart all server processes by executing the SYS$STARTUP:DQS$STARTUP.COM procedure.

When you redefine the DQS$IDLE_TIME logical name, con sider the following:

- If you define the idle time to be less than 15 minutes, processes may be created on your server more often than you would like.
- If you define the idle time to be greater than 15 minutes, too many processes may remain for too long on your server.

Example

Example 2.11 specifies that the server maintains logical links to idle clients for at least 45 minutes.

Example 2.11. Specifying the Duration of Links to Inactive Clients

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$IDLE_TIME "0 00:45:00.0" Return
```

Idle Time Exceeded

The DQS software responds as follows:

When . . .	Then . . .
Idle time specified by the DQS $IDLE_TIME logical name is exceeded	The server process disconnects the logical link to the client.
The client has a print job to transfer to the server	The client reestablishes the logical link to the server.

2.21. Move the Server Account Directory to Another Device

Storage Space

The amount of storage space required for the server account directory varies depending on the following:

- Number of queues defined
- Expected number and size of the queued print jobs
- Rate at which printers can print jobs (so that files are deleted)

The DQS server installation sets the default device for the server account to SYS$COMMON:. You can use the DQS$SERVER_CHANGE_DEFAULT_DEVICE.COM procedure to change the default to another disk or back to SYS$COMMON:.

Procedure

If you want to move the server account directory to a device other than the system disk, execute the DQS$SERVER_CHANGE_ DEFAULT_DEVIC E.COM procedure by entering the following command at the DCL prompt:

```
$ @SYS$MANAGER:DQS$SERVER_CHANGE_DEFAULT_DEVICE Return
```

DQS$SERVER_CHANGE_DEFAULT_DEVIC E.COM asks you for the device for the account storage.

```
Enter device for the DQS$SERVER account:
```

Enter the name of the device to use for the server account. For example, if your server account is to be on device USERDISK, enter USERDISK:.

```
Enter device for the DQS$SERVER account:USERDISK: Return
```

After you enter the device name, you see a series of status messages, such as the following:

```
User record(s) updated
DQS$SERVER directory is now located on device USERDISK:
To start DQS, enter following:
      $ @SYS$STARTUP:DQS$STARTUP.COM
In VMScluster environment, you need to start DQS on each node
in a cluster.
$
```

2.22. Advanced VMScluster DQS Configurations

In an OpenVMScluster environment, you can configure:

- A set of queues to be shared by all nodes in the cluster
- A set of queue to be shared by some nodes and a different set of queues to be shared by other nodes.

When you configure different sets of queues, you need to:

1. Redefine the DQS$NOTIFIER_LOCK logical
2. Specify two separate DQS$SERVER_CONFIG.TXT files.

The following sections explain how to configure different sets of queues.

2.23. Specifying Unique Notifier Lock Names in a VMScluster Environment

Requirement

When you configure different sets of queues for different groups of nodes, you must define the same notifier lock name on all cluster members that share a particular set of queues.

Procedure

To ensure that each set of queues is associated with a common and unique notifier lock, define the DQS $NOTIFY_LOCK logical name to be the same, unique lock name on all cluster nodes that share the same set of queues.

By default, the DQS installation names the notifier lock to be DQS$NOTIFIER.

Example

For example:

- A VMScluster system consists of nine nodes with four members that share one set of queues and five members that share another set of queues.
- Four members that share the same set of queues use the default notifier lock name DQS$NOTIFIER.
- Five cluster members that share the other set of queues require that you define a unique notifier lock name on each of them. The lock name DQS$NOTIFIER_2 can be defined on each of these five cluster members as shown in Example 2.12.

Example 2.12. Specifying Unique Notifier Lock Names in a Nonhomogeneous VMScluster

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$NOTIFY_LOCK DQS$NOTIFIER_2 Return
```

2.24. Redefining DQS $SERVER_CONFIG_DAT_FILE and DQS $SERVER_CONFIG_TXT_FILE

Recommended Configuration

VSI recommends that you use the same DQS server configuration for all nodes within your cluster. The DQS installation sets up the DQS software on your system with this recommended configuration in mind. As a result, the default values for the locations of the DQS server configuration files are both specifically located in the SYS$COMMON areas of your clusters file system.

The DQS software uses VMS logical names to locate the server configuration files.

Location of Configuration Data

The location of the server configuration text is controlled by the logical DQS $SERVER_CONFIG_TXT_FILE. The default value for this logical is SYS$COMMON:[SYSMGR]DQS$SERVER_CONFIG.TXT.

The location of the encoded file that the DQS$SERVER.EXE program uses to validate a client's request for access to a particular server queue is controlled by the logical DQS$SERVER_CONFIG_DAT_FILE. The default value for this logical is SYS$COMMON:[SYS EXE]DQS$SERVER_CONFIG.DAT.

Specifying Multiple Server Configuration Files

DQS can support multiple server configuration files within a cluster. If your cluster cannot use the default configuration, you can use the DQS$SERVER_CONFIG_TXT_FILE and DQS $SERVER_CONFIG_DAT_FILE logical names to provide multiple, separate, DQS server configurations within the same cluster.

Procedure

To modify the logicals, use the following procedure:

1. Edit the site-specific DQS startup file SYS$STARTUP:DQS$SYSTARTUP.COM.

2. Change SYS$COMMON: to SYS$SYSROOT: in following lines:

```
$ DEFINE/SYS/EXEC DQS$SERVER_CONFIG_TXT_FILE -
       SYS$COMMON:[SYSMGR]DQS$SERVER_CONFIG.TXT
$ DEFINE/SYS/EXEC DQS$SERVER_CONFIG_DAT_FILE -
       SYS$COMMON:[SYSMGR]DQS$SERVER_CONFIG.DAT
```

   Restart DQS to effect the change by executing SYS$STARTUP:DQS$STARTUP.COM.

2.25. Specifying Remote Queues on a VMScluster System

Execute the Start-up Procedure

If you specify remote queues on any node in a VMScluster system, you must execute the SYS $STARTUP:DQS$STARTUP.COM procedure on each cluster member. If you do not, DQS connections to the cluster fail randomly because not all nodes in the cluster are configured as DQS servers.

The cluster should have an incoming cluster alias defined so that clients can direct their print jobs to that cluster alias.

2.26. Configuring Daisy-Chained Queues

Benefits

Daisy-chained queues are useful when a server's output device becomes unavailable. In this case, you can define the remote queue for the unavailable device also to be a client queue that directs print jobs to yet another remote queue on another DQS server node.

Defining

To define a remote queue as a daisy-chained queue, define the remote queue to be a DQS client queue, using any of the methods described in Chapter 3:

- Use SYS$MANAGER:DQS$DEFINE.COM if you want the definition to be temporary.
- Add the definition to DQS$SYSTARTUP.COM if you want a permanent defintion.

Beware of Loops

The DQS software does not prevent you from creating a loop in your daisy-chained queues. Be sure not to set up daisy-chained queues where print jobs ultimately are directed back to the queue from which they were forwar ded.

Fixing Looping Errors

Perform the following procedure to eliminate a loop:

1. Stop any queue in the loop, then stop the other queues in the loop.

2. Edit the configuration file (DQS$SYSTARTUP.COM) and change the client and/or server queue definitions to eliminate the loop.

3. Restart DQS and the queues on each node. DQS$NOTIFIER deletes any copies of jobs that were created in the loop.

4. Enter the QSHOW command to verify that the loop has been eliminated.

5. Manually delete any print jobs that were replicated in the loop.

2.27. Summary of Server Tasks

Table 2.1 lists the management tasks and how the task is performed for the DQS server.

Table 2.1. QSET/ENTRY Command Qualifiers That Function Differently with DQSPrinting

For this task . . .	You do this . . .
Editing site-specific startup file to define logical names	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM procedure so that it defines each DQS logical name definition you specify.
Setting up remote queues	Add entries to the SYS$MANAGER:DQS $SERVER_CONFIG.TXT file. For each remote queue on the server node, include a QUEUE keyword followed by the name of the queue as the first argument and the name of the client node that is permitted to send jobs to the remote queue as the second argument.
Granting clients controlled access to a server	Add entries to the SYS$MANAGER:DQS $SERVER_CONFIG.TXT file. For each client node allowed access to remote queues on the server system, include the ALLOW_ NODE keyword with the name of the client node as the first and only argument. Client nodes must also be allowed access through the QUEUE entries in the server configuration file.
Denying clients access to a server	Add entries to the SYS$MANAGER:DQS $SERVER_CONFIG.TXT file. For each client node denied access to remote queues on the server system, include the DENY_NODE keyword with the name of the client node not permitted to access the server as the first and only argument.
Logging client access to a server	Define the DQS$LOG_ACCE SS logical name so that the server creates a file in the server account that has the name of a client and counts each time the client establishes a connection to the server.
Enabling client accounting information	Define the DQS$ACCOUNTING_BY_SYST EM logical name so that a client's node name (in the format *node-name*) is used in the server's accounting record.
Enabling status messages for a queue	Define the DQS$STATUS_*queue-name* logical name to be a status message that is displayed for each queue that has this logical name defined when a QSHOW command is issued.
Enabling the DQS print symbiont for a remote queue	Issue OpenVMS commands to stop the remote queue, initialize the queue specifying the DQS$PRTSMB process, and restart the queue.
Specifying the interval at which the server notification process scans the remote queues	Define the DQS$NOTIFY_CYCLE_TIME logical name to be the interval at which the DQS server notification process (DQS

For this task . . .	You do this . . .
	$NOTIFIER) scans the remote queues to determine when a print job has completed.
Specifying a directory for Print Server log files	Define the DQS$LOG_AREA logical name to be the directory into which Print Server log files are placed.
Specifying the priority of the server process	Define the DQS$PRIORITY logical name to be the priority at which you want the server process to run.
Specifying the maximum priority of DQS print jobs	Define the DQS$MAX_PRIORITY logical name to be the maximum priority at which a print job can be queued on the server.
Specifying the time for maintaining links to client nodes	Define the DQS$IDLE_TIME logical name to be the duration of time the server maintains a logical link to a client symbiont that is idle (that is, a symbiont that is not transferring print jobs).
Moving the DQS server default directory.	Run the DQS$SERVER_CHANGE_DEFAULT_ DEVICE.COM procedure.
Configuring different sets of queues in a VMScluster system.	Define the same DQS$NOTIFY_LOCK logical name on all nodes in a nonhomogeneous VMScluster that share the same queue file. By default, the DQS installation names the notifier lock to be DQS $NOTIFIER. Redefine the logicals DQS$SERVER_CONFIG_DAT_ FILE and DQS$SERVER_CONFIG_TXT_FILE.
Configuring server software in a VMScluster	Execute the DQS$STARTUP.COM command file on all cluster members in order to define remote queues throughout the cluster.
Setting up daisy-chained remote queues	Define a remote queue to also be a client queue that directs jobs to yet another remote queue.

Chapter 3. Configuring and Managing a DQS Client

This chapter presents an overview of the management tasks required by a DQS client and explains how to:

- Use the DQS startup procedure in client management
- How to manage processes created by client queues
- Create client queues with basic and advanced methods
- Temporarily change the definition of a client queue
- Delete client queues
- Configure client nodes in a VMScluster environment
- Synchronize client and server forms definitions
- Spool to a client queue
- Modify the default note on the banner page of a print job
- Regulate forms checking on client nodes

3.1. Overview of Client Management

Management Tasks

Configuring and managing a DQS client primarily involves defining DQS client queues to direct print jobs to associated remote queues on server systems. You can also modify and delete DQS client queue definitions.

DQS software provides the following methods of defining and managing client queues:

- A basic method using the DQS$IMPORT.COM procedure
- More complex methods:
    - Adding definitions as parameters to the DQS$DEFINE.COM procedure in the YS $MANAGER:DQS$SYSTARTUP.COM file
    - Defining generic client queues
    - Defining logical client queues

Executing the DQS Startup Procedure

To start or restart client queues, you must use the SYS$STARTUP:DQS$STARTUP.COM procedure. This procedure invokes the required procedures to define client queues on the system.

3.2. Managing Client Processes

Overview

On a client node, every client queue that you create (that is not a generic or a logical queue) results in the creation of one process.

One Client; One Process

If you do not anticipate a problem with the number of processes on your client node, then you can use either of the following procedures to create client queues:

- DQS$IMPORT.COM (described in the section titled Creating Client Queues with DQS $IMPORT.COM)
- DQS$DEFINE.COM (described in the section titled Creating Client Queues with More Advanced Methods)

Both result in the creation of a process for each client queue.

Reducing Processes

If you do not want to create as many processes as client queues, then you can create either of the following types of client queues:

- Generic client queues (described in the section titled Method 2: Defining Generic Client Queues)
- Logical client queues (as described in the section titled Method 3: Defining Logical Client Queues)

Generic and logical queues direct jobs to other client execution queues that actually process the job. Therefore, fewer processes are created on your client node.

3.3. Creating Client Queues with DQS $IMPORT.COM

A Basic Method

The most basic way to create DQS client queues is to invoke the DQS$IMPORT.COM comman d procedure.

The DQS$IMPORT.COM procedure provides a way to permanently define client queues. The queue definition is automatically added to the DQS$SYSTARTUP.COM comman d file. This ensures that the client queue starts with the proper definition each time the client system is rebooted.

Before You Start

Before you can define a client queue with the DQS$IMPORT command procedure:

- You must have the following privileges: SYSPRV, OPER, SYSNAM, NETMBX, and TMPMBX.
- The associated server node must be reachable.

- The associated remote queue must be initialized and defined as a DQS remote queue.

Procedure

Create a client queue with the DQS$IMPORT.COM procedure as follows:

Step	Action
1	Invoke the DQS$IMPORT.COM command file for each client queue you want to create. `$ @SYS$MANAGER:DQS$IMPORT server-node::remotequeue Return` For each client queue, supply the associated remote queue and server node in the format *server-node*::*remote-queue* as a parameter to the procedure.

DQS$IMPORT.COM then does the following:

- Creates the client queue, giving it the name of its associated remote queue.
- Adds the client queue definition to the DQS$SYSTARTUP.COM file so that the queue definition is persistent.

Example

Example 3.1 shows how to use DQS$IMPORT.COM to create a client queue Q_CHICAGO that uses the name of the correspon ding remote queue Q_CHICAGO on the server node APPLES.

Example 3.1. Creating a Client Queue with DQS$IMPORT.COM

```
$ @SYS$MANAGER:DQS$IMPORT APPLES::Q_CHICAGO Return
```

DQS$IMPORT.COM adds this client queue definition to the DQS$SYSTARTUP.COM file.

Note

DQS$IMPORT.COM adds an invocation of the DQS$DEFINE.COM procedure to the SYS $MANAGER:DQS$SYSTARTUP.COM file for each client queue and executes the DQS $DEFINE.COM procedure to actually create the queue. When you use DQS$IMPORT.COM to create a client queue, the client queue and the remote queue have the same name.

3.4. Creating Client Queues with More Advanced Methods

Advanced Methods

More advanced methods of creating client queues are as follows:

- Adding a definition for a client queue directly to the DQS$SYSTARTUP.COM file using the DQS $DEFINE.COM procedure and parameters
- Defining generic client queues
- Defining logical client queues

Method 1: Using DQS$DEFINE

Creating a client queue with the DQS$DEFINE.COM procedure is useful when you want the name of a client queue to be different from its associated remote queue name.

Procedure

Step	Action

1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file and search for the statement, "DQS Client Queues Definitions". The statement appears in the file as follows:
	
	`$!-----------------------------------------`
	`$! DQS Client Queues Definitions`
	`$!-----------------------------------------.`

2	On a new line following the "DQS Client Queue..." statement, insert a DCL command that invokes the SYS$MANAGER:DQS$DEFINE.COM procedure (see Example 3.2). For each invocation of DQS$DEFINE.COM that you insert, supply the following parameters in the following order (the last parameter is optional): a. Client queue name b. Its corresponding server node name c. Its corresponding remote queue name d. A zero (0) e. Qualifiers with which to start the client queue Separate each parameter by at least one space. Spaces are not permitted in the parameters. Example 3.2 shows how to edit this information properly into the SYS$MANAGER:DQS$SYSTARTUP.COM file.

DQS$DEFINE.COM is the procedure that actually creates a DQS client queue. When you edit the SYS$MANAGER:DQS$SYSTARTUP.COM file, you should insert your invocation of DQS$DEFINE.COM exactly as shown in Example 3.2.

Example

In Example 3.2, SYS$MANAGER:DQS$SYSTARTUP.COM is edited to contain an invocation of the DQS$DEFINE.COM procedure that associates the client queue CQ_ILLINOIS with the remote queue Q_CHICAGO on server node APPLE.

Example 3.2. Creating Client Queues With DQS$DEFINE.COM

```
$ EDIT/EDT SYS$MANAGER:DQS$SYSTARTUP.COM Return
     1      $! IDENT=V1.3
*FIND 'DQS Client Queues Definitions' Return
     113    $! DQS Client Queues Definitions
$!
*INSERT Return
```

```
    $ @SYS$MANAGER:DQS$DEFINE CQ_ILLINOIS -          ! Queue being defined
                APPLE -                              ! Server node
                Q_CHICAGO -                          ! Queue on server node
                0 -                                  ! Unit number of
 pseudo-device
                /NOENABLE_GENERIC                    ! Optional qualifiers
 on local queue
Ctrl/Z
      113    $! DQS Client Queue Definitions
*EXIT
SYS$COMMON:[SYSMGR]DQS$SYSTARTUP.COM;6   119 lines
$
```

Method 2: Defining Generic Client Queues

You can define client queues to be generic queues, which direct print jobs to execution queues on the client for processing. Defining generic queues gives you many independently targeted queues processed by a single symbiont. In this way, you trade off parallel job transfers for fewer processes on your client node.

Procedure

To define generic client queues, perform the following steps:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM. You should include all generic queue definitions in the SYS$MANAGER:DQS$SYSTARTUP.COM file so that they are permanent.

2	Add a definition of the execution queue. Your command should have the following format:
	
	`$ @SYS$MANAGER:DQS$DEFINE execution-queue -`
	`_$ server-node remote-queue`
	`_$ 0 /ENABLE_GENERIC[/qualifiers]`

3	Add the definition of the DQS$REMOTE_*client-queue* logical name for the associated server node and remote queue, in the form, *server-node*::*remote-queue*:
	
	`$ DEFINE/SYSTEM/EXECUTIVE_MODE -`
	`_$ DQS$REMOTE_client-queue server-node::remote-queue`

4	Add the command to initialize and start the queue specifying the /GENERIC qualifier. The parameter to the /GENERIC qualifier must be the name of the execution queue on the client system. Your command should have the following format:
	
	`$ INITIALIZE/QUEUE/START/GENERIC=execution-queue -`
	`_$ client-queue`

5	Save the DQS$SYSTARTUP.COM file and exit from the editor.
6	Execute the DQS$STARTUP.COM procedure.

If the DQS$REMOTE_*client-queue* logical name is not defined for a generic queue, the logical name definition for the execution queue is used to establish the associated server node and remote queue.

Example

Example 3.3 shows how to define a generic client queue.

Example 3.3. Defining Generic Client Queues

```
$ @SYS$MANAGER:DQS$DEFINE D NODEC QUEUE4 0 /ENABLE_GENERIC Return
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$REMOTE_A NODEA::QUEUE1 Return
$ INITIALIZE/QUEUE/START/GENERIC=D A Return
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$REMOTE_B NODEB::QUEUE2 Return
$ INITIALIZE/QUEUE/START/GENERIC=D B Return
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$REMOTE_C NODEA::QUEUE3 Return
$ INITIALIZE/QUEUE/START/GENERIC=D C Return
```

Example 3.3 defines the generic queues A, B, and C, which direct print jobs to the client execution queue D. There is a single symbiont running for queue D.

- If you print to queue A, your job transfers to server node NODEA, remote queue QUEUE1.
- If you print to queue B, your job transfers to server node NODEB, remote queue QUEUE2.
- If you print to queue C, your job transfers to server node NODEA, remote queue QUEUE3.
- If you print to queue D, your job transfers to server node NODEC, remote queue QUEUE4.

Efficiency Suggestion

An execution queue connects to the associated server for each of its generic queues. However, defining generic queues with the same associated server to direct jobs to the same execution queue is most efficient.

Refer to the *OpenVMS System Manager's Manual* for more information on generic queues.

Method 3: Defining Logical Client Queues

Logical queues are similar to generic queues in that each can have its own remote queue, independent of the execution queue. The difference is that you use the ASSIGN/QUEUE comman d to associate the logical queue with its execution queue.

The DQS software determines the associated server node and remote queue by looking for a DQS $REMOTE_*client-queue* logical name for each logical queue.

Procedure

To define logical client queues, perform the following steps:

Step	Action
1	Stop the client queue with the following command: `$ STOP/QUEUE/RESET client-queue Return`
2	Define the DQS$REMOTE_*client-queue* logical name for the associated server node and remote queue, in the form, *server-node*::*remote-queue*. Your command should have the following format: `$ DEFINE/SYSTEM/EXECUTIVE_MODE -`

Step	Action
	_$ **DQS$REMOTE_client-queue server-node::remote-queue Return**
3	Associate the logical client queue with a client execution queue. $ **ASSIGN/QUEUE execution-queue logical-queue Return**
4	Start the logical queue. $ **START/QUEUE logical-queue Return**

If DQS$REMOTE_*client-queue* is not defined for the logical queue, the DQS software uses the logical name definition for the execution queue.

Example

Example 3.4 shows how to define a logical queue.

Example 3.4. Defining Logical Client Queues

```
$ STOP/QUEUE/RESET QUEUE_A Return
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$REMOTE_QUEUE_A NODEX::QUEUE_Y Return
$ ASSIGN/QUEUE QUEUE_D QUEUE_A Return
$ START/QUEUE QUEUE_A Return
```

Example 3.4 assigns the logical client queue QUEUE_A to direct print jobs to the execution queue QUEUE_D, which is a DQS queue. QUEUE_D processes jobs queued to QUEUE_A and directs them to the server NODEX, remote queue QUEUE_Y.

Refer to the *OpenVMS System Manager's Manual* for more information on logical queues.

3.5. Temporarily Changing the Definition of a Client Queue

Overview

The DQS software determines a client queue's associated server node and remote queue by translating the DQS$REMOTE_*client-queue* logical name to be the associated server and remote queue. The DQS $REMOTE_*client-queue* logical name is defined for a client queue by both the DQS$IMPORT.COM and the DQS$DEFINE.COM procedures.

Procedure

To temporarily change the target of a client queue:

Step	Action
1	Stop the client queue by entering the following command at the DCL prompt: $ **STOP/QUEUE/RESET client-queue Return**
2	Redefine the DQS$REMOTE_*client-queue* logical name at the DCL prompt by entering: $ **DEFINE/SYSTEM/EXECUTIVE_MODE DQS$REMOTE_clientqueue server-node::remote-queue Return**
3	Restart the queue at the DCL prompt by entering:

Step	Action
	$ **START/QUEUE queue-name Return**

This temporary definition of the client queue is lost the next time the client system reboots because the logical name definition was not added to the SYS$MANAGER:DQS$SYSTARTUP.COM file.

Example

Example 3.5 shows how to change the definition of a client queue temporarily. This example stops the client queue CQ_ILLINOIS, redefines its remote node to be server node CHERRY, remote queue Q_OHIO, and restarts the client queue.

Example 3.5. Temporarily Changing the Remote Queue

```
$ STOP/QUEUE/RESET CQ_ILLINOIS Return
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$REMOTE_CQ_ILLINOIS CHERRY::Q_OHIO Return
$ START/QUEUE CQ_ILLINOIS Return
```

3.6. Deleting Client Queues

Procedure

To delete a client queue, perform the following steps:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file and delete the lines that define the client queue.
2	Stop the queue by entering the following command at the DCL prompt: $ **STOP/QUEUE/RESET client-queue Return**
3	Delete any jobs in the queue. $ **QDELETE/ENTRY=job-entry-numbers client-queue Return**
4	Delete the queue by entering the following command at the DCL prompt: $ **DELETE/QUEUE client-queue Return**
5	Deassign the queue logical name: $ **DEASSIGN/SYSTEM/USER DQS$REMOTE_queue-name Return** Where *queue-name* is the name of the client queue that you deleted.

3.7. Configuring Client Nodes in a VMScluster Environment

Execute Startup File

You must execute the SYS$STARTUP:DQS$STARTUP.COM command file on each cluster member to define the required DQS logical names on each cluster member. This action makes available the DQS software to users in the cluster regardless of the cluster node.

Cluster Alias

You should use an outgoing cluster alias as the name of your client system. This makes the cluster appear to be one system to its server(s). If you do not define and use this cluster alias, users may switch cluster members and may not be able to show, modify, or delete print jobs that they own.

3.8. Synchronizing Client and Server Forms Definitions

About the DQS$CLIENT Program

You should synchronize form definitions between your client and server systems. Do this by running the DQS client program SYS$SYSTEM:DQS$CLIENT.EXE. This requires OPER, TMPMBX, and NETMBX privileges.

The DQS$CLIENT program looks at the queues on a client. For each queue, it queries the associated server(s) to determine what forms are defined there. DQS$CLIENT then defines all the server forms on the client system.

Procedure

To synchronize forms on your DQS client and server systems:

Step	Action
1	Run the DQS$CLIENT program by issuing the following command: `$ RUN SYS$SYSTEM:DQS$CLIENT Return`

DQS$CLIENT synchronizes client forms with the server forms in alphabetical order of the client queue names. Each server is queried only once, regardless of the number of client queues associated with remote queues on that server.

Conflicting Server Forms Definitions

If DQS$CLIENT finds conflicting form definitions on different servers, it is not able to define both forms on a DQS client. For example, if one DQS server defines form 3 as X and another defines 3 as Y, form 3 can be defined as either X or Y at a DQS client. No practical way exists to determine which case of form 3, X or Y, occurs at the client.

If DQS$CLIENT encounter s more than one form number for a given form name, you may not get the results you desire.

Form Definition Override

For DQS$CLIENT to override the local form definition with its value on the server, you must define the DQS$FORM_ OVERRIDE logical name on the client system to be an arbitrary value.

If you do not define DQS$FORM_OVERRIDE on your client node, your local form definitions are maintained.

When you run DQS$CLIENT, you are notified that your local form definition is superseded by the server's definition (with DQS$FORM_OVERRIDE defined) or remains the same (DQS $FORM_OVERRIDE not defined).

Procedure

To allow the server form definition to supersede the client definition:

Step	Action
1	Edit SYS$MANAGER:DQS$SYSTARTUP.COM file and search for the following statement: `$! Form Synchronization:`

2	Uncomment the DQS$FORM_OVERRIDE logical name with an arbitrary value, for example, TRUE.
	
	`$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$FORM_OVERRIDE`
	`_$ TRUE Return`

3	Exit from the DQS$SYSTARTUP.COM file.

3.9. Spooling to a Client Queue

Procedure

To spool to a DQS client queue, you must associate a LAT port with the DQS client queue. To do so, perform the following steps:

Step	Action
1	Run the LATCP utility.
2	Create the LAT port LTA*nnn*:. The *nnn* represents a number that is not already in use on your system, for example, LTA999:.
3	Exit from LATCP utility.
4	Set that LAT port device spooled.

A user-written program can then output to the specified LAT port device. The program output is written to a disk and queued to the specified output queue when the file is closed.

Note

If the LAT software is not started, enter the following command at the OpenVMS DCL prompt:

```
$ @SYS$STARTUP:LAT$STARTUP.COM Return
```

Example

Example 3.6 shows how to set up a LAT port device.

Example 3.6. Setting Up a LAT Port Device

```
$ MCR LATCP Return
LATCP> CREATE PORT LTA999: Return
LATCP> SHOW PORT Return
Port Name   Port Type      Status       Remote Target (Node/Port/Service)
```

```
--------- ----------- -------------
 ----------------------------------------
_LTA999:    Application    Inactive     //
LATCP> EXIT Return
$ SET DEVICE LTA999:/SPOOLED=QUEUE_Z Return
```

You can spool this LTA999 device to only one DQS client queue at a time. Example 3.7 shows how to change the DQS client queue that is spooled to the LTA9999: device, by entering the following DCL commands where *new_queue_name* is the name of the new DQS client queue:

Example 3.7. Changing Client Queue Spooled to LTA999: Device

```
$ SET DEVICE LTA999/NOSPOOLED Return
$ SET DEVICE LTA999/SPOOLED=new_queue_name Return
```

Example 3.8 shows how to use the LAT port with a FORTRAN program. In this example, the FOR $PRINT logical is associated with the LTA999: device. In this case, the LTA999: device is spooled to a DQS client queue QUEUE_Z (see Example 3.6). When the program runs, the output is directed through the FOR$PRINT logical to the LTA999: device, and from there to the spooled queue, QUEUE_Z.

Example 3.8. Spooling to a LAT Port Device

```
$ DEFINE FOR$PRINT LTA999: Return
$ SHOW QUEUE QUEUE_Z Return
     Server queue QUEUE_Z, stopped, on YOUNG::, mounted from DEFAULT
$ TYPE EXAMPLE.FOR Return
           print *,'This is a test.'
           end
$ RUN EXAMPLE Return
$ SHOW QUEUE QUEUE_Z Return
Server queue    QUEUE_Z,    stopped,    on YOUNG::,    mounted from DEFAULT
Jobname    Username    Entry    Blocks    Status
-------    --------    -----    ------    ------
FORPRINT   YOUNG         573         1     Pending
FORPRINT   YOUNG         573         1    Pending
$
```

This example shows QUEUE_Z as stopped to show the submitted job in the DQS server queue.

3.10. Controlling the Note on the Job Banner Page

How the Note is Printed

The DQS software makes use of a print attribute to place a note on the banner page of a print job when logical names are defined in the DQS$SYSTARTUP.COM file. On the client node, this logical name definition provides the default text "Print Job transported by DQS V*n.n*". The note defined by this logical name prints on the banner page of a print job if the /NOTE qualifier is not specified by the client node user on the PRINT command.

The note that prints on the banner page of a print job follows this order of precedence:

- Note specified by the user with PRINT/NOTE
- Note specified by the client logical name definition (DQS$CLIENT_DEFAULT_J OB_NOTE)

- Note specified by the server logical name definition (DQS$SERVER_DEFAULT_J OB_NOTE)

Options

You have the following options:

- Use the default note provided by the DQS software for the banner page.
- Turn off the DQS logical definitions for the banner page note.
- Define your own note for the banner page.

To Turn Off the Default

To turn off the DQS default note on the client system, do the following:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.
2	Search for the following command and remove the exclamation mark (!) comment character: `$! $ DEASSIGN/SYSTEM/EXEC DQS$CLIENT_DEFAULT_JOB_NOTE`
3	Save the file SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.

The change takes efect the next time the SYS$STARTUP:DQS$STARTUP.COM is executed.

To Change Note Text

To change the text of the note that prints on the banner page, do the following:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file.

2	Search for the following command:
	
	`$! $ DEFINE/SYSTEM/EXEC DQS$CLIENT_DEFAULT_JOB_NOTE -`
	`$!  "Place your text here"`
	
	Substitute your note for the text “Place your text here” and remove the dollar sign and exclamation mark ($!) at the beginning of each line.

3	Save the file SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.

The next time the SYS$STARTUP:DQS$STARTUP.COM is executed and the note is not specified by the user, your customized note prints on the job banner page.

3.11. Regulating Forms Checking

The Forms Change Timer

A part of the DQS software known as the forms change timer periodically checks DQS client queues to see if an operator or the software, itself, has properly changed forms.

The DQS software requires a match of form types between the one mounted on a DQS client queue and the one specified for each of the queue’s jobs.

Logical Names

Regulating the forms change timer component requires editing the SYS$MANAGER:DQS $SYSTARTUP.COM file and adding logical name definitions:

Table 3.1. QSET/ENTRY Command Qualifiers That Function Differently with DQSPrinting

Qualifier	Function Without DQS	Function With DQS
DQS $NO_FORMS	TRUE	Turns off the forms timer on all DQS queues. If a DQS client node submits a job to one of its queues and the job requires a form that is not mounted on the queue, no data transfers, no errors occur, and the job waits indefinitely.
DQS $FORM_*queue_name*	TRUE	Names a particular DQS client queue, *queue_name*. You can repeat the logical definition for any number of DQS client queues. If a DQS client node submits a job to the named queue with a form that is not mounted on the queue, the data transfers and no errors occur. The result is similar to having the form mounted on the queue.

Procedure

To regulate forms checking, do the following:

Step	Action
1	Edit the SYS$MANAGER:DQS$SYSTARTUP.COM file to add the logical names. Look for the following statement: `$! Forms Change Timer:`
2	Uncomment the command that defines the logical name DQS$NO_FORMS to be TRUE for all DQS client queues: `$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$NO_FORMS "TRUE"`
3	Uncomment the command that defines a logical name DQS$FORM_*queue_name* to be TRUE, where *queue_name* is a particular DQS client queue. `$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$FORM_queue_name "TRUE"` This command applies only to the named DQS client queue. Repeat this command for any number of DQS client queues.
4	Save the file, SYS$MANAGER:DQS$SYSTARTUP.COM and exit from the editor.
5	Execute the SYS$STARTUP:DQS$STARTUP.COM procedure: `$ @SYS$STARTUP:DQS$STARTUP.COM Return`

Example

Example 3.9 shows how to add logical names to regulate forms checking on VMScluster systems with 50 clients or more. This examples turns off the forms change timer for all client queues and allows client

queues LNO6_ROOM23A, LPS17_NUM016, and LPS17_600_BLUE to accept print jobs regardless of the forms mounted on them.

Example 3.9. Regulating Forms Checking on Many Client Systems

```
$!
$!     Forms Change Timer:
$!
$!     Uncomment the following logical if you desire to turn off the forms
$!     change timer. Digital recommends that you do this on VMScluster
$!     systems with 50 or more DQS client queues.
$!
! DEFINE/SYSTEM/EXECUTIVE_MODE DQS$NO_FORMS TRUE
$!
$! Ignoring forms definitions on specific client queues:
$!
$!   You can disable certain client queues from rejecting print jobs when
$!   the form mounted on the client queue differs from the form mounted
$!   on the remote queue. Define a logical of the form DQS
$FORM_queue_name .
$!   This allows the client queue, 'queue_name' , to accept and process
 jobs,
$!   regardless of the form mounted on the remote queue. For example:
$!
$!
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$FORM_LNO6_ROOM23A "TRUE"
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$FORM_LPS17_NUM016 "TRUE"
$ DEFINE/SYSTEM/EXECUTIVE_MODE DQS$FORM_LPS17_600_BLUE "TRUE"
```

After adding these logical name definitions to the site-specific startup file DQS$SYSMANAGER:DQS $SYSTARTUP.COM and executing the DQS startup procedure SYS$STARTUP:DQS$STARTUP, the DQS software no longer checks for forms mismatches between each named queue and its jobs.

Chapter 4. Troubleshooting

This chapter provides some general guidelines for troubleshooting apparent problems with the DQS software and suggests recovery procedures when:

- Remote notification is not working
- Jobs remain in the remote queue with completion status
- A job is not printing
- Large numbers of DQS client queues slow down the network

The guidelines are not meant to cover all possible error conditions, but are intended to help you understand some common situations.

4.1. General Troubleshooting Guidelines

What To Do First

If something fails to work with the DQS software, check these first:

- Determine whether your DQS configuration has changed recently.
- Run the DQS Installation Verification Procedure (IVP) located in SYS$TEST:DQS$IVP.COM. It may confirm that the DQS software is functioning properly, or it may detect the error condition.
- Check that the DQS$STARTUP.COM procedure is invoked in the system start-up file,
  - SYS$STARTUP:SYSTARTUP_V5.COM for OpenVMS VAX V5.*n* systems
  - SYS$STARTUP:SYSTARTUP_VMS.COM for OpenVMS VAX V6.*n* systems and OpenVMS AXP systems

  DQS$STARTUP.COM starts the DQS software and invokes procedures that define the required DQS logical names. If the DQS$VERSION system logical name is not defined, then this may be the cause of your error condition.
- On client nodes, check that queues are defined in the SYS$MANAGER:DQS$SYSTARTUP.COM file.
- On server nodes, check that remote queues are set up in the SYS$MANAGER:DQS $SERVER_CONFIG.TXT file.

4.2. Running the IVP

Command

You can execute the Installation Verification Procedure (IVP) to check if the DQS software is running properly. To run the IVP, enter:

```
$ @SYS$TEST:DQS$IVP.COM Return
```

IVP on a Client Node

Following is a sample of an IVP run on a node with the DQS client software configured.

```
DQS V1.3 Installation Verification Procedure

Restricted Rights: Use, duplication, or disclosure by the U.S. Government
is subject to restrictions as set forth in subparagraph (c)(1)(ii) of
DFARS 252.227-7013, or in FAR 52.227-19, or in FAR 52.227-14 Alt. III, as
applicable.

This software is proprietary to and embodies the confidential technology
of Digital Equipment Corporation. Possession, use, or copying of this
software and media is authorized only pursuant to a valid written license
from Digital or an authorized sublicensor.
```

For a client-only node, the following prompt is displayed:

```
It is necessary to test the DQS client with a remote DQS server node.
Enter remote server NODE::QUEUE:
```

Enter the name of a server node and a remote queue on that node. For example, to test the proper operation of server queue Q_CHICAGO on node APPLE, enter the following:

```
Enter remote server NODE::QUEUE APPLES::Q_CHICAGO RETURN
```

For the IVP to succeed on a client-only installation, the following must exist:

- A remote queue on the server node that is accessible from the client system (see the chapter on configuring a DQS server in the Chapter 2 for information on making server queues accessible).
- The DQS$IVP_TEST_FORM on the server node. If this form no longer exists on the server node, enter the following DCL commands on the server node:

  ```
  $ DEFINE/FORM DQS$IVP_TEST_FORM 1110 -
  /STOCK = DQS_IVP_TEST_STOCK -
  /DESCRIPTION = "DQS test form, do not delete" Return
  ```

Then, if the DQS software is working properly. the IVP displays a series of messages (this may take several minutes), such as:

```
%DQS-I-CRESRVQUE Creating remote queue
%DQS-I-CRECLIQUE Creating client queue
%DQS-I-CHKCLIQUE Checking client queue
%DQS-I-ENTJOB Entering test job in client queue
%DQS-I-CHECKJOB Checking test job in client queue
%DQS-I-SNDJOB Sending job to remote queue
%DQS-I-CHKSRVQUE Checking test job in remote queue
%DQS-I-DELJOB Deleting job from remote queue
%DQS-I-CHKDEL Checking that job is gone from remote queue
%DQS-I-CLEANUP Cleaning up, deleting IVP queues
%DQS-I-QUESUCCESS IVP test successful
```

IVP on a Server Node

Following is a sample of an IVP run on a node with the DQS server software configured.

```
DQS V1.3 Installation Verification Procedure
```

```
Restricted Rights: Use, duplication, or disclosure by the U.S. Government
is subject to restrictions as set forth in subparagraph (c)(1)(ii) of
DFARS 252.227-7013, or in FAR 52.227-19, or in FAR 52.227-14 Alt. III, as
applicable.

This software is proprietary to and embodies the confidential technology
of Digital Equipment Corporation. Possession, use, or copying of this
software and media is authorized only pursuant to a valid written license
from Digital or an authorized sublicensor.
```

Then, if the DQS software is working properly. the IVP displays a series of messages (this may take several minutes), such as:

```
%DQS-I-CRESRVQUE Creating remote queue
%DQS-I-CRECLIQUE Creating client queue
%DQS-I-CHKCLIQUE Checking client queue
%DQS-I-ENTJOB Entering test job in client queue
%DQS-I-CHECKJOB Checking test job in client queue
%DQS-I-SNDJOB Sending job to remote queue
%DQS-I-CHKSRVQUE Checking test job in remote queue
%DQS-I-DELJOB Deleting job from remote queue
%DQS-I-CHKDEL Checking that job is gone from remote queue
%DQS-I-CLEANUP Cleaning up, deleting IVP queues
%DQS-I-QUESUCCESS IVP test successful
```

If the DQS IVP encountered problems, the procedure displays error messages. Refer to Appendix A, DQS System Manager Messages for an explanation of specific error messages.

4.3. Problem: Remote Notification Is Not Working for Any Jobs

Action

If remote notification is not working check for the following:

- The DQS$NOTIFIER process is running on the server.
- The network object DQS is defined on the client.
- The client node is defined in the server node's NCP database for DECnet Phase IV systems or the server node's NCL database for DECnet/OSI Phase V systems.
- If the client node is in a cluster, make sure that the DQS software is properly configured on all nodes of the cluster.

4.4. Problem: Notification of Print Job Completion Is Not Occuring for Some Jobs

Action

The DQS$NOTIFIER searches the node for all known DQS server queues when it starts. If you define a new queue as a valid DQS server queue, you must stop and restart the DQS$NOTIFIER process to

enable print job com pletion messages of jobs submitted to the new queue. The simpliest way to do this is to restart DQS.

If the DQS$NOTIFIER is not restarted:

- Print jobs remain in the DQS server queue with the retained attribute set.
- The temporary files are not deleted.
- The user is not notified when the job is completed.

4.5. Problem: Jobs Remain in the Remote Queue With Completion Status

Action

- Make sure that the DQS$NOTIFIER process is running on the server.
- If the server is a member of a VMScluster system in which the queues are not common across the cluster members, check the definition of the DQS$NOTIFY_LOCK logical name as described in Chapter 2.

4.6. Problem: A Job Is Not Printing

Information You Need

To determine why a print job does not appear to be printing, you need to determine the following for the job:

- The status; is it in the process of printing on the server node? Is it stopped or stalled?
- The location; is it in the client queue or the remote queue?

Using the QSHOW Command

To obtain this information, issue a QSHOW command at the client system of the client queue to which the job was queued for printing. The output of the QSHOW command provides information on both the status and the location of the print job.

Examining the QSHOW Display

The output of the QSHOW command is designed to provide all the information you need to determine the status and location of a print job. The command output usually appears as follows, showing the location of the job on either the client or the server system:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
   Jobname    Username    Entry    Blocks    Status
   -------    --------    -----    ------    ------
   JOB        GUENTHER     2737         1    Printing
```

Understanding Location Information

Note that the client queue and remote queue are referred to in the command output as the "Server queue" and the "Remote queue," respectively. This terminology is tied to the OpenVMS concept of served queues. For the DQS QSHOW command,

- Server queue refers to the DQS client queue
- Remote queue refers to the DQS remote queue

When the QSHOW command output contains information for both the client and remote queues (as shown in the preceding display) you should first decide whether the job is in the client queue, in the remote queue, or neither.

Location: Neither Node

If the job is not in either the client or the remote queue, then the following possibilities exist:

- The job printed and is lost.
- The user printed to a different queue or printed a different job.
- The actual printer was different from the expected one. The system manager or the DQS software may have rerouted the request to another available printer.

If you suspect a DQS problem, examine the NETSERVER.LOG files in the DQS server account's directory or, if OpenVMS accounting is enabled, check the accounting files.

Location: Server Node

If the job is located on the server node, then the DQS software has functioned properly. If the job still is not printing, then the following possibilities exist:

- It may be pending because of form or characteristic problems.
- It may be held or waiting to print after a specified time.

Check that the printer is on line, powered on, and has paper in it.

Location: Client Node

If the job is still located on the client node, then examine the status of the client queue in the QSHOW output. The queue status is either stopped or stalled.

Understanding Status Information

You see the following status if there is a problem:

- **Stopped**: If the queue is stopped, the QSHOW command output contains the reason for stopping the queue if the DQS software stopped the queue. The queue may be stopped because the remote system is unknown or because the client node is denied access to the server.

  Check the OPCOM output or the OPERATOR.LOG file for messages regarding why the queue stopped.

- **Stalled**: Stalled means that the DQS symbiont has not sent a checkpoint message to the job controller within the timeout limit of the stall timer. This could mean that the server is running

slowly. It could also mean that the DQS symbiont failed to connect to the server and is in the process of retrying the connect.

Check for the OPCOM message "Remote system off line." Disk problems on the server can also cause a stalled state.

The stalled state is transient and requires no special attention. The DQS software automatically tries the job every five minutes. Generally, this action alone corrects the stalled state.

Any of the following indicate that the client queue is operational and running:

- **Printing**: Indicates that the job is transferring to the server node. This may signify that the user was impatient, that the server system was overloaded, or that the transfer is slow. This status is shown as follows:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       MACOMBER    2738        1   Printing
```

- **Pending**: Indicates that the job cannot be transferred for some reason local to the client system. You can determine the reasons for this pending state by entering:

```
$ QSHOW/FULL queue-name Return
```

Possible reasons for a pending status are the following:

- A characteristic was used to print the job that is not enabled for the queue.

  In this case, add the appropriate characteristics to the queue or remove them from the print job.

- The form type mounted for the queue may not match the form type specified for the job.

  In this case, the DQS software automatically changes forms.

  This status is shown as follows:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       DAVIES      2739        1   Pending
```

- **Holding**: Indicates that the job cannot be transferred because the user specified that the job should be held. This status is shown as follows:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       WOODWARD    2740        1   Holding
```

- **Holding until**: Indicates that the job cannot be transferred because the user specified that the job should be held until after the specified time. This status is shown as follows:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       SMITH       2741        1   Holding until
```

- **Retained on completion**: Indicates that the job has transferred to the server system. This status is shown as follows:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       SCHULLMAN   2742        1   Retained on completion
```

- **Retained on error**: Indicates that the job failed to transfer to the server system because of the error shown with the job.

  The user should correct the problem and retry the request. This status is shown as follows:

```
** Remote queue - [ CQ_EUROPE => Q_PARIS, on CHERRY:: ]
Terminal queue Q_PARIS
============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       TIMLEGE     2743        1   Retained on error
  %JBC-E-NOSUCHFORM, no such form
```

The Remote Queue Display

Issuing a QSHOW comman d for a client queue sometimes does not show job status information because either the remote queue or the server node is inaccessible. In these cases, you see the following in the QSHOW command output:

- No remote queue information
- A message explaining why the remote queue is not reachable

Missing Remote Queue Information

When the remote queue information does not display when you issue a QSHOW comman d, it means that the DQS software does not view the client queue as a valid DQS queue. The command output would appear as follows:

```
Server queue CQ_EUROPE, mounted form DEFAULT
Jobname   Username   Entry   Blocks   Status
-------   --------   -----   ------   ------
JOB       CHO         2744        1   Pending
```

In this case, you should issue a QSHOW/FULL comman d for the queue. If the command output contains the information /PROCESSOR=DQS$SMB, invoke the DQS$STARTUP.COM file to define the queue as a valid DQS queue.

You can also check the DQS$REMOTE_*client-queue* logical name definition for the client queue. It should be defined system wide and in executive mode to be the corresponding server node and remote queue.

Error Messages in the Remote Queue Display

In some cases, the QSHOW command output contains an error message that cites a reason why the remote queue is not accessible. The command output would appear as follows:

```
***** Remote queue CHERRY::Q_PARIS not accessible *****
      %SYSTEM-F-UNREACHABLE, remote node is not currently reachable
=============================to be transferred======================
Server queue CQ_EUROPE, mounted form DEFAULT
```

The following is a list of error messages that might be displayed when a remote queue is inaccessible:

- -DQS-F-BADNODE, specified node has been denied access to this server.

  ```
  %DQS-W-MSG_REMOTE, remote system error:
  -DQS-F-BADNODE, specified node has been denied access to this server.
  %SYSTEM-F-LINKDISCON, network partner disconnected logical link
  ```

  Explanation: The specified queue is not accessible with the DQS software because the client system has been specifically denied access to the server.

  User Action: Check with the server system manager to remove the client's node name from deny status in the DQS$SERVER_CONFIG.TXT file.

- -DQS-F-BADQUE, specified queue not valid for DQS access.

  ```
  %DQS-W-MSG_REMOTE, remote system error:
  -DQS-F-BADQUE, specified queue not valid for DQS
  access.
  =============================to be transferred======================
  Server queue CQ_EUROPE, mounted form DEFAULT
  ```

  Explanation: The specified queue is not accessible with the DQS software. Either the client's node name was not added to the access list or the remote queue does not exist.

  User Action: Check with the server system manager to determine the reason for the error.

- %SYSTEM-F-INVLOGIN, login information invalid at remote node

  ```
  ***** Remote queue CHERRY::Q_PARIS not accessible *****
  %SYSTEM-F-INVLOGIN, login information invalid at remote node
  =============================to be transferred======================
  Server queue CQ_EUROPE, mounted form DEFAULT
  ```

  Explanation: The server account information may not be defined properly.

  User Action: Make sure the server's password in the network object database on the server matches the password in the UAF. Also, make sure the owner and UIC of the server's directory is the server's account.

- -DQS-E-NOTSTARTED, DQS has not been started yet

```
%DQS-W-MSG_REMOTE, Remote system error:
-DQS-E-NOTSTARTED, DQS has not been started yet
==============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
```

  Explanation: The DQS software was not started on the server.

  User Action: Ask the server system manager to start the DQS software. After the DQS software is started, there may be a delay of up to five minutes before jobs are transferred.

- %SYSTEM-F-NOSUCHNODE, remote node is unknown

```
***** Remote queue CHERRY::Q_PARIS not accessible *****
%SYSTEM-F-NOSUCHNODE, remote node is unknown
==============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
```

  Explanation: The server node is not known to your client node.

  User Action: Either add the server node to your DECnet node database or change the node name to a known node.

- %SYSTEM-F-NOSUCHOBJ, network object is unknown at remote node

```
***** Remote queue CHERRY::Q_PARIS not accessible *****
%SYSTEM-F-NOSUCHOBJ, network object is unknown at remote node
==============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
```

  Explanation: Either the specified node is not a DQS server node or the DQS server has not been started.

  User Action: Change the node to a DQS server or start the DQS server on the server node.

- %SYSTEM-F-UNREACHABLE, remote node is not currently reachable

```
***** Remote queue CHERRY::Q_PARIS not accessible *****
%SYSTEM-F-UNREACHABLE, remote node is not currently reachable
==============================to be transferred==================
Server queue CQ_EUROPE, mounted form DEFAULT
```

  Explanation: The server is not on the network. If the job is in the queue, it is transferred to the server soon after the server becomes reachable.

  User Action: Wait until the server becomes reachable or queue your print job to a remote queue on another server node.

4.7. Problem: Many DQS Client Queues Slow Down the Network

About the Problem

A part of the DQS software known as the forms change timer periodically checks DQS client queues to see if an operator or the software has properly changed forms. The DQS software requires a match

of form types between the one mounted on a DQS client queue and the one specified for each of the queue's jobs. This checking can slow down the overall performance of a VMScluster system with 50 or more DQS client queues.

Action

To increase performance of systems with many DQS client queues, you can turn off the forms change timer component of the DQS software. As a result, the software cuts forms-checking operations to a minimum and performance is improved. VSI recommends that you do this on VMScluster systems with 50 or more DQS client queues.

Regulating the forms change timer component requires editing the SYS$MANAGER:DQS $SYSTARTUP.COM file and adding logical name definitions DQS$NO_FORMS and DQS$FORM_ *queue_name*. For information, refer to the section titled

Regulating Forms Checking in Chapter 3.

Appendix A. DQS System Manager Messages

This appendix alphabetically lists and defines all error and diagnostic messages issued by DQS. Where appropriate, it describes any corrective actions that you can take.

DQS messages have the same format as standard OpenVMS system messages and appear as follows:

```
%DQS-l-ident, text
```

Where

- *l* is the severity level of the message, such as -W-, for a warning message.
- *ident* is the message identification, such as NOACCESS.
- *text* is the message text, such as "specified node does not have access to this server".

DQS messages use the same severity codes as OpenVMS system messages: S, I, W, E, and F.

DQS messages that concern printing with the DQS software are documented in Appendix A in *Distributed Queuing Service for OpenVMS Systems User's Guide* for the user and in this appendix for the system manager.

Refer to the *OpenVMS System Messages and Recovery Procedures Reference Manual* for an explanation of the format and a definition of severity codes for OpenVMS system messages.

A.1. Submitting a Software Performance Report

When to Submit

When you receive a DQS internal software error, you should submit a Software Performance Report (SPR) to VSI.

Refer to the Appendix C in *Distributed Queuing Service for OpenVMS Systems Installation Guide* for information on how to submit an SPR.

A.2. Message Section

DQS Notification Messages

The DQS software can return the following notification messages (these messages do not follow the standard OpenVMS message format).

Print job *job* (queue *queue*, entry *job-number*) completed at *date time*

Explanation: Your print job has printed successfully on the server.

User Action: None. This is an informational message.

Print job *job* (queue *queue*, entry *job-number*) terminated with error status at *date time*

Explanation: Your print job has not printed successfully on the server node. A secondary error message is displayed that describes the reason why the error occurred.

User Action: Rectify the problem based on the information in the secondary message.

A.2.1. DQS Message Definitions

Because all DQS messages have the same prefix %DQS- *l*-, the messages in this section are listed alphabetically according to their message *ident*.

BADFILE, UL total syntax errors detected in file: *filename errors*.

Explanation: The DQS$SERVER_UPDATE_CONFIG.EXE program detected errors when arsing the server configuration text file. The name of the file and the total number of syntax errors in the file are displayed.

User Action: Edit the file and fix the syntax errors.

Invoke the SYS$MANAGER:DQS$SERVER_UPDATE_CONFIG.COM command procedure to effect the change.

BADFORM, INTERNAL SOFTWARE ERROR - Translate form failed for form *form-name*.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

BADNAM, INTERNAL SOFTWARE ERROR - process name does not contain underscore, and must.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

BADNODE, specified node has been denied access to this server.

Explanation: Your client is denied access to the server to which you are attempting to queue your print job.

User Action: Consult the server system manager and request that your client node have access to the server. The server system manager needs to modify the DQS$SERVER_CONFIG.TXT file to allow this access.

BADQUE, specified queue not valid for DQS access.

Explanation: The remote queue you specified is not defined on the server as accessible to DQS clients. This means one of the following conditions exists:

- The SYS$MANAGER:DQS$SYSTARTUP.COM comman d file on the server is not set up correctly.
- The SYS$STARTUP:DQS$STARTUP.COM file has not been executed.
- The client is not permitted to use the remote queue.

User Action: Consult the server system manager and request that your node be granted access to the remote queue. The system manager may need to modify the DQS$SERVER_CONGIG.TXT file on the server node.

BAD_REMOTE_NAME, *logical-name* must be defined for this queue.

Explanation: The client queue you specified has not been defined as a DQS client queue. For a queue to be defined as a client queue, a logical name of the form DQS$REMOTE_*client-queue* must be defined for the queue.

User Action: Check the SYS$MANAGER:DQS$SYSTARTUP.COM comman d file to see whether the client queue has been defined; if so, reexecute the SYS$STARTUP:DQS$STARTUP.COM command file. If not, add a command line to the SYS$MANAGER:DQS$SYSTARTUP.COM comman d file that invokes the SYS$MANAGER:DQS$DEFINE.COM command file and defines the queue as a DQS client queue. Then reexecute SYS$STARTUP:DQS$STARTUP.COM.

If the queue is not intended to be a DQS client queue, you can initialize the queue with a symbiont other than the DQS symbiont.

If these actions fail to solve the problem, submit an SPR.

BADTMPFILE, bad temporary file.

Explanation: A temporary file created during the IVP could not be deleted.

User Action: The IVP may not have failed. Delete the file manually.

BLKCNTEXC, INTERNAL SOFTWARE ERROR - Block count in message header too large.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

CANTPRINT, cannot print job.

Explanation: The IVP could not execute the PRINT command and has failed.

User Action: Reinstall the DQS software.

CHANGEFORMFAILED, INTERNAL SOFTWARE ERROR - Change form failed for queue *queue-name*.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

CHECKPOINTED, job was checkpointed.

Explanation: Your print job was checkpointed and the DQS software cannot remove the checkpoint.

User Action: Delete your job from the queue and resubmit it.

DISKFULL, remote server disk is full.

Explanation: Your print job cannot be copied to the server node. Either there is not enough disk space on the server to store your file(s) for printing, or the quota for the DQS server account [DQS$SERVER] was exceeded.

User Action: If this error persists, either ask the server system manager to free up some disk space on the server, or queue your job to another server.

EMPTYFILE, DQS server configuration file: *filename* is empty.

Explanation: Either the DQS$SERVER_UPDATE_CONFIG.EXE program or the DQS$SERVER program encountered an empty file when trying to read the indicated file.

User Action: Check that the file exists, is unprotected, and is not corrupted. The file's contents may have somehow become corru pted. If file name is DQS$SERVER_CONFIG.DAT, you can regenerate the file by deleting DQS$SERVER_CONFIG.DAT and then invoking the SYS$MANAGER:DQS $SERVER_UPDATE_CONFIG.COM command procedure to recreate the file.

ENTRY_REQUIRED, entry=number required.

Explanation: You did not specify the required /ENTRY=*number* qualifier in your QSET or QDELETE command.

User Action: Reenter the command using the /ENTRY= *number* qualifier, specifying the appropriate job entry number.

ERROROPENING, error opening *file-name* as input.

Explanation: The DQS client symbiont cannot open the specified file for the reason cited in the secondary error message that accompanies this message. The secondary error message is printed in the print output on the server. Usually, this error message occurs when the file to be printed is deleted before the file is transferred by the client symbiont to the server.

User Action: Rectify the problem based on the information in the secondary message.

ERRORREADING, error reading *file-name*.

Explanation: The DQS client symbiont could not read the specified file for the reason cited in the secondary error message that accompanies this message.

User Action: Rectify the problem based on the information in the secondary message.

INACCESSIBLE, remote queue *queue-name* is inaccessible.

Explanation: A connection cannot be established to the server with which you are attempting to perform a QSHOW, QSET/ENTRY, or QDELETE/ENTRY function.

User Action: Remember that if you submit a print request to the server, it is not processed until the server is reachable. However, for QSHOW, QSET/ENTRY, or QDELETE/ENTRY command functions, the server must be reachable when the command is issued.

INCOMPPROT, incom patible DQS Server and Client protocol versions.

Explanation: The version of DQS software on the client is incom patible with the version of DQS software on the server.

User Action: Make sure that compatible versions of DQS software are installed on both the client and server systems.

INVALIDSERVER, invalid server node or queue name.

Explanation: During a client IVP, you entered either a node name or a queue name that is not valid. You are prompted to provide another node and queue name.

User Action: Enter a valid node and queue name in response to the prompt:

```
Enter remote NODE::QUEUE
```

INVBLKTYP, INTERNAL SOFTWARE ERROR - Invalid block type *number* in message.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

INVLINE, invalid line encountered in file: *filename*, at line number *line*.

Explanation: The DQS$SERVER_UPDATE_CONFIG.EXE program detected a syntax error when parsing the server configuration file. The name of the file and the line number where the error was detected are displayed.

User Action: Edit the file and fix the syntax error. Invoke the SYS$MANAGER:DQS $SERVER_UPDATE_CONFIG.COM comman d procedure to effect the change.

INVLOGFIL, invalid log file.

Explanation: You specified a log file incorrectly in a QSET /ENTRY command. The reason for the error is cited in the secondary error message that accompanies this message.

User Action: Correct the file specification and reenter the QSET/ENTRY command.

INVMSGLEN, INTERNAL SOFTWARE ERROR - Invalid message length in header.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

INVMSGTYP, INTERNAL SOFTWARE ERROR - Invalid message type *number* in header.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

INVMSG, INTERNAL SOFTWARE ERROR -invalid message in current state.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

INVQUAVAL, value '*value*' invalid for */qualifier-name* qualifier.

Explanation: The value you specified for the qualifier */qualifier-name* is invalid.

User Action: Consult the documentation if necessary and specify the proper value for the qualifier.

INVREQCOD, invalid request code, code =*code*.

Explanation: The job controller issued an invalid request code.

User Action: Submit an SPR.

INVVER, versions of protocol do not match.

Explanation: The software version of DQS on your client is incompatible with the software version of DQS on the server to which you are attempting to queue your print job.

User Action: Make sure that the proper versions of DQS software are installed both on your client and on your client's server(s).

JOBDELETED, *number* job(s) deleted.

Explanation: The QDELETE/ENTRY comman d you issued did not delete all of the jobs you specified, only the number of jobs specified by *number*.

User Action: Check the job number(s) for the job(s) you attempted to delete as well as your access privileges for deleting those jobs.

LONGNODENAME, Node name size problem due to DECnet /OSI fullname.

Explanation: The DQS symbiont received a node name (possibly from a daisy-chained queue) that is too long for the version of the OpenVMS operating system or the DECnet networking software to handle.

User Action: Upgrade systems to DECnet/OSI Phase V software or reconfigure queues or both.

MSG_PE, protocol error.

Explanation: A protocol error indication was received. This error is usually incurred by having incompatible DQS software versions on server and client systems.

User Action: Make sure that your client and server systems have compatible versions of DQS software installed on them.

MSG_REMOTE, remote system error.

Explanation: An error occurred on the remote system. The error is described in the secondary error message that accompanies this message.

User Action: Rectify the problem based on the information in the secondary message.

MSG_TYPE_RANGE, INTERNAL SOFTWARE ERROR - Message type received '*message-value*' out of range.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

MSG_TYPE_UNEXPECTED, INTERNAL SOFTWARE ERROR- Message type received '*message-value*' unexpected.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

MSG_UNK, INTERNAL SOFTWARE ERROR - Unknown error message class: *class*.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

NOACCESS, specified node does not have access to this server.

Explanation: Your client is not authorized to use the queue(s) on the server.

User Action: If you want your client to use the remote queue(s) on the server, you must have the server system manager grant your client system access to the server.

NODELETE, job cannot be deleted from queue.

Explanation: The IVP could not delete the test job from the remote queue and failed.

User Action: Delete the job manually, check the network link, and rerun the IVP.

NODELETE, /NODELETE is allowed only on local jobs.

Explanation: You cannot use the /NODELETE qualifier for a job that is not on your local node (that is, a job that has transferred to the server).

User Action: Do not use the /NODELETE qualifier for jobs that have transferred to the server.

NODEOFF, node *node-name* offline.

Explanation: The specified server is not currently reachable and a job is waiting to transfer to it.

User Action: Check the status of the server or ask the server system manager to bring the server back on line.

NOJID, INTERNAL SOFTWARE ERROR - Job number, job name, or * required.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

NOJOBDELETED, no jobs deleted.

Explanation: The QDELETE/ENTRY comman d you issued did not delete any jobs.

User Action: Check the job number of the job you are attempting to delete, as well your access privileges to delete the job.

NOJOBSET, no jobs set.

Explanation: The QSET/ENTRY command you issued did not set any jobs.

User Action: Check the job number of the job you are attempting to modify, as well your access privileges to modify the job.

NOPRIV, need SYSPRV, SYSNAM, OPER, NETMBX, TMBMBX privileges.

Explanation: Your account does not have the correct privileges, or they have not been enabled.

User Action: Change your account registration (or enable your privileges) and rerun the IVP.

NOREMOTEQUE, remote queue does not exist or is inaccessible.

Explanation: The remote queue you specified during the installation does not exist, the network link is down, or the client does not have access to the remote queue.

User Action: For a server IVP, reinstall the DQS software For a client IVP, check the target queue name, the network link, and that the DQS$QUEUE_ *queue-name* logical name on the server allows the client access. Then reinstall the server software and rerun the IVP.

NOSUCHQUE, no such remote queue *queue-name*.

Explanation: The remote queue specified by *queue-name* does not currently exist on the server, even though it is defined as a valid DQS remote queue.

User Action: Initialize the specified remote queue on the server.

NOTOWN, not owner of the job.

Explanation: You are not the owner of the job that you are attempting to delete or modify. The DQS software determines job ownership by the node name and user name with which the job was initiated.

User Action: Check to see that you have specified the correct job number. If so, make sure that you are working from the same account on the same client as when you initiated the job.

NOTSTARTED, The DQS software has not been started yet.

Explanation: The DQS software has not been started on the server yet.

User Action: Ask the server system manager to start the DQS software. Print jobs transfer up to 5 minutes after the software starts.

NO_ACCESS_CONTROL, no access control strings allowed in server node name.

Explanation: A DQS$REMOTE_*client-queue* logical name, which specifies a client queue's associated server and remote queue, includes access control information in its definition.

User Action: Redefine the logical name using the format DQS$REMOTE_*client-queue server-node*::*remote-queue*. Also, check the definition of the DQS$REMOTE logical name for the client queue in the DQS$STARTUP.COM command file.

NO_NODE_SPECIFIED, printer queue name does not specify a node.

Explanation: A DQS$REMOTE_*client-queue* logical name does not specify a node name in its definition.

User Action: Redefine the logical name using the format DQS$REMOTE_*client-node server-node*::*remote-queue*. Also, check the definition of the DQS$REMOTE logical name for the client queue in the DQS$STARTUP.COM command file.

NO_SET_LISTS, list of job numbers for SET not supported.

Explanation: You cannot specify a sequence of job numbers in a QSET/ENTRY command, such as /ENTRY=(1,2,3).

User Action: Issue a separate QSET/ENTRY command for each job entry.

OLDIVP, old IVP job exists in queue, deleting.

Explanation: An IVP test print job already exists in the queue. It is deleted.

User Action: No user action. This is an informational message.

OPENFILE, error opening DQS server configuration file: *filename*.

Explanation: Either the DQS$SERVER_UPDATE_CONFIG.EXE program or the DQS$SERVER program encountered an error when trying to open the indicated file.

User Action: Check that the file exists and is unprotected. Make sure there is sufficient disk space to create a new file.

P8RESERVED, parameter P8 is reserved for the DQS software on remote systems.

Explanation: You specified parameter 8 when issuing a PRINT command to a DQS queue. Parameter 8 is reserved for DQS information.

User Action: Do not specify parameter 8 when using the DQS software. Combine parameter 8 arguments with parameter 1 to parameter 7 arguments.

PE_MSG_NAME, INTERNAL SOFTWARE ERROR - Protocol error, message_type = *type*.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

PE_MSG_NUMBER, INTERNAL SOFTWARE ERROR - Protocol error, unknown message_type = # *number*.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

PROTOCOL_ERROR, INTERNAL SOFTWARE ERROR - Protocol error; invalid message received - *number*.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

READFILE, error reading DQS server configuration file: *filename*.

Explanation: Either the DQS$SERVER_UPDATE_CONFIG.EXE program or the DQS$SERVER program encountered an error when trying to read the indicated file.

User Action: Check that the file exists, is unprotected, and is not corrupted. The file's contents may have somehow become corru pted. If file name is DQS$SERVER_CONFIG.DAT, you can regenerate the file by deleting DQS$SERVER_CONFIG.DAT and then invoking the SYS$MANAGER:DQS $SERVER_UPDATE_CONFIG.COM command procedure to recreate the file.

REC_MSG, received message: "*message*".

Explanation: One of the previous messages came from a server that is running obsolete DQS software and that does not return a full error code.

User Action: Install the current version of DQS software on the server.

REINSTALL, install new DQS software.

Explanation: The version of DQS software on the client is incom patible with the version of DQS software on the server.

User Action: Make sure that your client and server systems have compatible versions of DQS software installed.

REPORT, reporting the following message to the client.

Explanation: The subsequent error messages are being reported to the client system.

User Action: None.

REQNOEXIST, requeue, remote queue does not exist on remote system.

Explanation: The queue you specified with a QSET/ENTRY/REQUEUE=*queue-name* command does not exist. You can only requeue to a queue that is on the same remote system as the print job.

User Action: Requeue your job to another remote queue, or delete the job and submit a new job to a valid remote queue.

SHUTDOWN, shutting down queue *queue-name* because of error.

Explanation: An error occurred while processing a print job on the queue *queue-name*. The queue is stopped because it cannot process any jobs.

User Action: Issue a QSHOW command of the specified queue to determine the reason for the error. Correct the error and restart the queue.

TOOMANYERRORS, too many errors talking to the server.

Explanation: Too many errors have occurred while attempting to transfer a print job to a server. The client symbiont stops until the problem is fixed.

User Action: Check the reported errors, rectify the problem, and restart the client queue.

UNEXPECTEDSTATE, INTERNAL SOFTWARE ERROR - Unexpected state: UL, encountered while parsing arguments.

Explanation: The DQS$SERVER_UPDATE_CONFIG.EXE program detected an inconsistency in its internal state while parsing the user input file.

User Action: Submit an SPR. If necessary, use the previous version of DQS$SERVER_CONFIG_.TXT file and rerun the program. [En

UNKNOWN_FUNCTION, INTERNAL SOFTWARE ERROR - function: *function-name* unknown.

Explanation: This is an internal DQS error.

User Action: Submit an SPR.

VALID_MESSAGES, INTERNAL SOFTWARE ERROR - Valid messages mask = *number*.

Explanation: Indicates which messages this server expected.

User Action: Submit an SPR.

WAIT, waiting for file to transfer to remote queue.

Explanation: The test file is being transferred or will be transferred.

User Action: No user action. This is an informational message.

WRITEFILE, error writing DQS server configuration file: *filename*.

Explanation: The DQS$SERVER_UPDATE_CONFIG.EXE program encountered an error when trying to write to the indicated file.

User Action: Check that the file exists and is unprotected. Make sure sufficient disk space exists to extend the file. When the problem is fixed, invoke the

SYS$MANAGER:DQS$SERVER_UPDATE_CONFIG.COM command procedure to effect the change.

XFERFAILED, Failed to transfer job to remote queue.

Explanation: The file could not be transferred to the remote queue.

User Action: An additional status message accompanies this error message to explain why the transfer failed.

Possible reasons for the transfer failure include the following:

- DQS$IVP_TEST_FORM may no longer exist on the server node.

  ```
  %JBC-E-NOSUCHFORM\ No such form
  ```

  To define the DQS$IVP_TEST_FORM, enter the following DCL command on the server system:

  ```
  $ DEFINE/FORM DQS$IVP_TEST_FORM 1110 -
  /STOCK = DQS_IVP_TEST_STOCK -
  /DESCRIPTION = "DQS test form, do not delete"
  ```

  Define this form on the server and start the IVP again.

- Not enough disk space is available, the DQS software was not started, or some other problem exists.

  ```
  %DQS-W-MSG-REMOTE, reason
  ```

  The *reason* defines the problem. Examine the reason, correct the problem, and try again.

XFERFAILED, transfer to server failed with error.

Explanation: Your current print job failed to transfer to the server node for the reason specified in the secondary error message. The DQS software attempts to retransfer the job after a wait.

User Action: If the problem persists, ask the server system manager to rectify the problem on the server node. The problem may be transient, such as unavailability of a temporary resource.

XFERTIME, timed out trying to transfer file to remote queue.

Explanation: The file transfer could not complete in 15 minutes.

User Action: Check the network link and rerun the IVP.

Appendix B. Using DQS With Various Symbionts

This appendix describes how jobs print when print symbionts other than the DQS print symbiont (DQS $PRTSMB) are used to process DQS print jobs.

Any symbiont that is built according to the rules specified in the OpenVMS documentation should be able to process DQS print jobs.

B.1. How Various Symbionts Function

Replacing the OpenVMS Symbiont

Only the OpenVMS print symbiont PRTSMB can be replaced with the DQS print symbiont DQS $PRTSMB. Replacing PRTSMB with the DQS print symbiont causes client-specific job information to be printed on the flag, burst, and trailer pages of DQS print output. This is recommended because the job information includes the client node job number, the client node name, and the time the print job is queued on the client node.

Other symbionts cannot be replaced with the DQS print symbiont. In these cases, the server job number and the time that the print job was queued on the server are printed on the output.

List of Print Symbionts

The following is a list of various print symbionts:

- **Standard OpenVMS print symbiont (PRTSMB)**. The DQS print symbiont should replace the OpenVMS symbiont.
- **PrintServer 40 symbiont (LP S$SMB)**. The LPS$SMB symbiont cannot be replaced by the DQS print symbiont.
- **LAT printer symbiont (LATSYM)**. The LATSYM symbiont cannot be replaced by the DQS print symbiont.
- **User-modified symbionts**. User-modified symbionts cannot be replaced by the DQS print symbiont. However, users can modify their symbionts to print client-specific information, as described in the section titled Modifying Your Symbiont To Output Client Job Information.
- **User-written symbionts**. User-written symbionts cannot be replaced with the DQS print symbiont. However, users can modify their symbionts to print client-specific information, as described in the section titled Modifying Your Symbiont To Output Client Job Information.

B.2. Modifying Your Symbiont To Output Client Job Information

DQS remote user information is stored in parameter 8. You can modify your symbiont to use this client-specific print information. The user information has the following format:

```
DQS: nnnnxjob-owner dd-mmm-yyyy hh:mm
```

Where:

- *nnnn* is the job number on the client node. This requires a minimum of 4 characters and a maximum of 10. If the number is less than 4 characters, the job number contains leading spaces.
- *x* is a single nonnumeric code that delimits the job number.
- *job-owner* is the client's node name (which may be trimmed) a double colon (::), and the client's user name.
- *dd-mmm-yyyy hh:mm* is the date and time the job was queued on the client.

Example

In this example of a remote user name, each circumflex character (^) denotes a single space.

```
DQS:^^^12xAPPLES::SHANNON^^^^05-FEB-1994^04:55
```

Important Parsing Information

Additional considerations for using parameter 8 include the following:

- If a parameter 8 string does not parse according to the format as specified, it is not processed as a P8 string.
- When interpreting the *job-number* field, if the value of the *job-number* is zero, the symbiont uses the local job number associated with the print job by the batch/print system. (This works because the value 0 is currently an illegal job number.)
- When processing the *job-owner* field, a symbiont should treat the entire *job-owner* as a string. The string is terminated by the first space. On OpenVMS systems, the *job-owner* for DQS purposes has the form NODE::USERNAME.

  Because of restrictions with trimmed full node names, a symbiont should not rely on the NODE-NAME::USERNAME type syntax, nor should a symbiont interpret the contents of the *job-owner* field.
- The P8 string has a maximum length of 254 characters (not including the null terminator).
- VSI reserves the right to add new information to the end of the P8 string. Print symbionts that parse this string should not depend on the string length nor should they interpret undocumented information that may appear in a particular P8 string.

Differences from Previous Versions

Item	In Version 1.2	In Version 1.3
Job Number field	Always 4 characters long.	May be more than 4 characters long.
Job Owner field	Number of spaces after the field varies.	Always one space after the field.

Appendix C. Server and Client System Logical Names

This appendix alphabetically summarizes the DQS server and client system logical names. Each logical name is described briefly, along with its value, function, default value (if any), and image.

C.1. DQS Product Logical Names

Important

The following logical names are used by the DQS software and should NOT be modified. These logical name definitions may affect the DQS client software, the DQS server software, or both.

Product Logical Name	Description, Value, Default, and Image
DQS$CONFIGURATION	**Description**: Type of DQS software installed. **Value**: Value is either CLIENT or SERVER. **Default**: None. **Image**: DQS$SERVER, DQS$SMB
DQS $CONTROLLED_ACCESS	**Description**: Set by the DQS software. Indicates on the server whether or not to check the security database to grant clients access to the server. **Default**: None. If this logical name is not defined on a server, the DQS software does not check on whether a client is a valid client of a server. **Image**: DQS$SERVER
DQS$DENY_ACCESS	**Description**: Set by the DQS software. Indicates on the server whether to check the security database to determine if clients have been denied access to the server. **Default**: None. All client nodes are valid. **Image**: DQS$SERVER
DQS$QUEUE_*remote-queue*	**Description**: Set by the DQS software. Identifies a queue as a DQS network-accessible queue. **Default**: None. If this logical name is not defined for a queue, the queue is not available as a DQS remote queue. **Image**: DQS$SERVER
DQS$VERSION	**Description**: A string that names the version number of the installed DQS software. **Value**: V*n.n*. The *n.n* is a number that stands for the currently installed version; for example, V1.3. **Default**: Version number of the installed product.

C.2. DQS Server Logical Names

Modifying Names

DQS software provides the following logical name definitions for the DQS server software. The system manager can modify the definitions in the site-specific startup procedure SYS$MANAGER:DQS $SYSTARTUP.COM.

Server Logical Name	Description, Value, Default, and Image
DQS $ACCOUNTING_BY_SYSTEM	**Description**: Enables client node names to be used in the account field in a server's accounting record. **Value**: Arbitrary. **Default**: If this logical name is not defined, accounting by client system is not enabled. **Image**: DQS$SERVER
DQS$IDLE_TIME	**Description**: The delta time that a DQS$SERVER process maintains a connection to an inactive client that is not sending print jobs. **Value**: The delta time. **Default**: "0 00:15:00.0" (15 minutes) **Image**: DQS$SERVER
DQS$LOG_ACCESS	**Description**: Creates a file in the [DQS$SERVER.NODES] subdirectory of the DQS server account each time a client establishes a connection to the server. **Value**: Arbitrary. **Default**: If this logical name is not defined, a file is not created in the [DQS$SERVER.NODES] directory. **Image**: DQS$SERVER
DQS$LOG_AREA	**Description**: Specifies the directory in which PrintServer log files are placed. **Value**: The name of the directory. **Default**: The DQS server account directory, [DQS$SERVER]. **Image**: DQS$NOTIFIER
DQS$MAX_PRIORITY	**Description**: Specifies the maximum processing priority with which a user can have a job queued on a server. **Value**: The maximum priority. **Default**: The value of the SYSGEN parameter DEFQUEPRI, which is usually 100.

Server Logical Name	Description, Value, Default, and Image
	Image: DQS$SERVER
DQS$NOTIFY_CYCLE_TIME	**Description**: Interval at which the server process scans remote queues to determine when print jobs are complete. **Value**: Time in OpenVMS delta time. **Default**: "0 00:05:00.0" (5 minutes) **Image**: DQS$NOTIFIER
DQS$NOTIFY_LOCK	**Description**: Prevents multiple DQS$NOTIFIER processes from running on a homogeneous VMScluster system. **Value**: The name of the lock. **Default**: DQS$NOTIFIER **Image**: DQS$NOTIFIER
DQS$PRIORITY	**Description**: Priority at which the server process runs. **Value**: Form decimal number between 0 and 16. **Default**: Priority 4. **Image**: DQS$SERVER
DQS $SERVER_CONFIG_TXT_FILE	**Description**: Specifies the location of the DQS $SERVER_CONFIG.TXT file. **Value**: File specification **Default**: SYS$COMMON:[SYSMGR]DQS $SERVER_CONFIG.TXT **Image**: DQS$SERVER_UPDATE_CONFIG.EXE
DQS $SERVER_CONFIG_DAT_FILE	**Description**: Specifies the location of the DQS $SERVER_CONFIG.DAT file. **Value**: File specification **Default**: SYS$COMMON:[SYSMGR]DQS $SERVER_CONFIG.DAT **Image**: DQS$SERVER_UPDATE_CONFIG.EXE, DQS $SERVER
DQS $SERVER_DEFAULT_JOB_NOTE	**Description**: Places a note on the banner page of DQS jobs to indicate that the DQS software was used to print the job. **Value**: Text string that is the message to print on the banner page of print jobs. **Default**: Text string "Print Job processed by DQS V1.3." **Image**: DQS$SERVER

Server Logical Name	Description, Value, Default, and Image
DQS$STATUS_*queue-name*	**Description**: Specifies that a status message for a queue is displayed when a QSHOW command of the queue is issued. **Value**: Arbitrary text string that is the status message to be displayed. **Default**: If this logical name is not defined, no status message is displayed. **Image**: DQS$CLIENT, DQS$SMB, DQS$SERVER

C.3. Client Logical Names

Modifying Names

DQS software provides the following logical name definitions for the DQS client software. These definitions can be modified by the system manager in the site-specific startup procedure SYS $MANAGER:DQS$SYSTARTUP.COM.

Client Logical Name	Description, Value, Default, and Image
DQS $CLIENT_DEFAULT_JOB_NOTE	**Description**: Places a note on the banner page of DQS jobs to indicate that the DQS software was used to print the job. **Value**: Text string that is the message to print on the banner page of print jobs. **Default**: Text string "Print Job transported by DQS V1.3." **Image**: DQS$SMB
DQS$FORM_OVERRIDE	**Description**: Overrides a client's form definition with its server's definition. **Value**: Arbitrary. **Default**: None. **Image**: DQS$CLIENT
DQS$REMOTE_*client-queue*	**Description**: Associates the client queue with a corresponding server system (or with multiple servers) and with a remote queue on each server. **Value**: The server node name and remote queue name in the format server-node::remote-queue. **Default**: None. This logical name must be defined in order for the queue to function as a DQS client queue. **Image**: DQSSMB, DQSCLIENT
DQS$STATUS_*queue-name*	**Description**: Specifies that a status message for a queue is displayed when a QSHOW command of the queue is issued.

Client Logical Name	Description, Value, Default, and Image
	Value: Arbitrary text string that is the status message to be displayed. **Default**: If this logical name is not defined, no status message is displayed. **Image**: DQS$CLIENT, DQS$SMB, DQS$SERVER