OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems

OpenVMS

Part Number: AA–PV5NA–TK

OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems

Order Number: AA–PV5NA–TK

May 1993

This manual is part two of a task-oriented guide to managing an OpenVMS system.

Revision/Update Information: This is a new manual.

Software Version: OpenVMS AXP Version 1.5
OpenVMS VAX Version 6.0

Digital Equipment Corporation
Maynard, Massachusetts

May 1993

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK6016

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

15 Managing System Page, Swap, and Dump Files

16 Performance Considerations

18 Tracking Resource Use

19 VMScluster Considerations

22 Managing the LAT Software

23 Managing DECdtm Services

24 Managing Special Processing Environments

A Files–11 Disk Structure

Glossary

Index

Examples

Figures

Tables

Preface

Intended Audience

The intended audience for this manual is OpenVMS system managers.

Document Structure

The *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems* consists of the following chapters:

- Chapter 14, Managing System Parameters
- Chapter 15, Managing System Page, Swap, and Dump Files
- Chapter 16, Performance Considerations
- Chapter 17, Getting Information About the System
- Chapter 18, Tracking Resource Use
- Chapter 19, VMScluster Considerations
- Chapter 20, Network Considerations
- Chapter 21, Managing InfoServer Systems
- Chapter 22, Managing the LAT Software
- Chapter 23, Managing DECdtm Services
- Chapter 24, Managing Special Processing Environments
- Appendix A, Files–11 Disk Structure
- Glossary

For more information about the structure of the *OpenVMS System Manager's Manual*, see Section 1.1.

Associated Documents

You will find the following books helpful when used in conjunction with the *OpenVMS System Manager's Manual*:

- *OpenVMS System Management Utilities Reference Manual*
- *OpenVMS DCL Dictionary*
- *OpenVMS User's Manual*
- *OpenVMS Software Overview*
- The current version of the *Upgrade and Installation Manual* for your system
- *OpenVMS VAX Guide to System Security*
- *OpenVMS AXP Guide to System Security*

- *Guide to OpenVMS Performance Management*
- *VMScluster Systems for OpenVMS*
- The manuals in the networking kit of the OpenVMS Standard Documentation Set:
  - *DECnet for OpenVMS Guide to Networking*
  - *DECnet for OpenVMS Networking Manual*
  - *DECnet for OpenVMS Network Management Utilities*

Conventions

In this manual, every use of OpenVMS AXP means the OpenVMS AXP operating system, every use of OpenVMS VAX means the OpenVMS VAX operating system, and every use of OpenVMS means both the OpenVMS AXP operating system and the OpenVMS VAX operating system.

The contents of the display examples for certain utility commands described in this manual may differ slightly from the actual output provided by these commands on your system. However, when the behavior of a command differs significantly between OpenVMS VAX and OpenVMS AXP, that behavior is described in text and rendered, as appropriate, in separate examples.

The following conventions are used to identify information specific to OpenVMS AXP or to OpenVMS VAX:

AXP	The AXP icon denotes the beginning of information specific to OpenVMS AXP.
VAX	The VAX icon denotes the beginning of information specific to OpenVMS VAX.
♦	The diamond symbol denotes the end of a section of information specific to OpenVMS AXP or to OpenVMS VAX.

The following conventions are also used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
GOLD *x*	A sequence such as GOLD *x* indicates that you must first press and release the key defined GOLD, then press and release another key. GOLD key sequences can also have a slash (/), dash (-), or underscore (_) as a delimiter in EVE commands.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)

. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in a VMS file specification, or in the syntax of a substring specification in an assignment statement.)
{ }	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in Bookreader versions of the manual.
italic text	Italic text emphasizes important information, indicates variables, and indicates complete titles of manuals. Italic text also represents information that can vary in system messages (for example, Internal error *number*), command lines (for example, /PRODUCER=*name*), and command parameters in text.
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
-	A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.
numbers	All numbers in text are assumed to be decimal, unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

14
Managing System Parameters

When your system is installed or upgraded, values of system parameters are automatically set by the command procedure SYS$UPDATE:AUTOGEN.COM (AUTOGEN), which is supplied by Digital. Digital recommends you use AUTOGEN regularly to adjust the values for system parameters to fit your hardware configuration and your system's work load.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Converting your customized parameter settings for use with AUTOGEN	Section 14.3
Modifying system parameter values with AUTOGEN (recommended method)	Section 14.5
Controlling AUTOGEN's parameter settings with MODPARAMS.DAT	Section 14.5.1
Automating AUTOGEN reports	Section 14.6
Managing system parameters with SYSMAN	Section 14.7
Managing system parameters with SYSGEN	Section 14.8
Managing system parameters with a conversational boot	Section 14.9

This chapter explains the following concepts:

Concept	Section
System parameters	Section 14.1
Default, current, and active values	Section 14.1.1
Pages and pagelets	Section 14.1.2
The recommended method for changing system parameter values	Section 14.2
The AUTOGEN command procedure	Section 14.4
AUTOGEN feedback	Section 14.4.1
The AUTOGEN feedback report (AGEN$PARAMS.REPORT)	Section 14.4.2
AUTOGEN phases	Section 14.4.3
The AUTOGEN parameter file (MODPARAMS.DAT)	Section 14.4.4

14.1 Understanding System Parameters

The system uses values for **system parameters** to control how the system functions. System parameters control a wide range of system functions, including but not limited to the following:

- Memory management

- Scheduling
- Security attributes
- Windowing system choice
- Terminal configuration
- VAXcluster or VMScluster attributes

The *OpenVMS System Management Utilities Reference Manual* lists and describes each system parameter.

Your distribution kit provides **default values** for system parameters to allow you to boot any supported configuration. When your system is installed or upgraded, a command procedure supplied by Digital (SYS$UPDATE:AUTOGEN.COM) executes to evaluate your hardware configuration, estimate typical work loads, and adjust the values of system parameters as needed.

Each system parameter has associated minimum and maximum values that define the scope of allowable values.

Parameter Types

System parameters can be one or more of the following types:

Type	Description
Dynamic	The value of a dynamic system parameter can be modified while the system is active by changing the *active* value in memory. In contrast, if you change the value of a parameter that is not dynamic, you must change the *current* value stored in the parameter file, and you must reboot the system for the changed value to take effect. For information on active and current values, see Section 14.1.1.
General	The value of a general parameter affects the creation and initialization of data structures at boot time.
Major	Major parameters are most likely to require modification.
Special	Special parameters are intended for use only by Digital. Change these parameters only if recommended by Digital personnel or in the installation guide or release notes of a Digital-supplied layered product.

Parameter Categories by Function

System parameters can be divided into the following categories, according to their function:

Category	Function
ACP	Parameters associated with file system caches and Files–11 XQP (extended QIO procedure) or ancillary control processes (ACPs).[1]
Cluster	Parameters that affect VAXcluster or VMScluster operation.
Job	Parameters that control jobs.
LGI	Parameters that affect login security.
Multiprocessing	Parameters associated with symmetric multiprocessing.

[1]Many ACP parameters are applicable only when Files–11 On-Disk Structure Level 1 disks are mounted or when an ACP is specifically requested during a mount command. In versions of the operating system before Version 4.0, a separate process, the Ancillary Control Process (ACP), performed file operations such as file opens, closes, and window turns. Version 4.0 introduced the XQP (extended QIO procedure), which allows every process on the system to perform these operations. For compatibility reasons, the names of the parameters have not changed.

Category	Function
PQL	Parameters associated with process creation limits and quotas.
RMS	Parameters associated with OpenVMS Record Management Services (RMS).
SCS	Parameters that control system communication services (SCS) and port driver operation. The parameters that affect SCS operation have the prefix SCS.
SYS	Parameters that affect overall system operation.
TTY	Parameters associated with terminal behavior.
User-defined	The following parameters can be user-defined: USERD1 (dynamic) USERD2 (dynamic) USER3 USER4

14.1.1 Default, Current, and Active Values

A system has several different sets of values for system parameters. The following table describes these values:

Value	Description
Default values	Values provided with the system to allow you to boot any supported configuration.
Current values	Values stored in the default parameter file on disk and used to boot the system. On VAX systems, the default parameter file is VAXVMSSYS.PAR. On AXP systems, the default parameter file is ALPHAVMSSYS.PAR.
Active values	Values that are stored in memory and are used while the system is running. You can change the active value on a running system only for system parameters categorized as dynamic system parameters.
Values stored in other parameter files	For special purposes, you can create a parameter file other than the default parameter file that is used to store current values.

When the system boots, it reads the current values into memory, creating active values. An active value remains equal to the current value until you change either the active value or the current value.

When you execute the AUTOGEN command procedure through the SETPARAMS phase, it changes *current* values.

The System Management utility (SYSMAN) and the System Generation utility (SYSGEN) allow you to show and modify both *current* and *active* values. You use the USE and WRITE commands to specify which values you want to show or modify.

14.1.2 Pages and Pagelets

VAX

On VAX systems, the operating system allocates and deallocates memory for processes in units called **pages**. A page on a VAX system is 512 bytes. Some system parameter values are allocated in units of pages. ♦

AXP

On AXP systems, some system parameter values are allocated in units of pages, while others are allocated in units of **pagelets**.

A page on an AXP system can be 8 kilobytes (KB) (8192 bytes), 16 KB, 32 KB, or 64 KB. A pagelet is a 512-byte unit of memory. One AXP pagelet is the same size as one VAX page. On an AXP 8KB computer, 16 AXP pagelets equal one AXP page.

When reviewing parameter values, especially those parameters related to memory management, be sure to note the units required for each parameter. Section 14.7.2 and Section 14.8.2 explain how to show parameter values and their units of allocation. ♦

14.2 Recommended Method for Changing Parameter Values

Many system parameters can affect other parameters and the performance of the system. For this reason, Digital recommends you use the Digital-supplied command procedure SYS$UPDATE:AUTOGEN.COM (AUTOGEN) to manage system parameters. For information on AUTOGEN, see Section 14.4.

The System Management utility (SYSMAN) and the System Generation utility (SYSGEN) also allow you to manage system parameters. Although these utilities are not generally recommended for *changing* parameter values, you might want to use one of these utilities for the following reasons:

- To display system parameters and their values
- To temporarily modify a single parameter that has little effect on other parameters

Caution

If you change a parameter value with SYSMAN or SYSGEN, the value you set will be overridden or reset to the default value when you run AUTOGEN. To ensure that parameter changes are retained when you run AUTOGEN, you must add the parameter value to the AUTOGEN parameter file MODPARAMS.DAT. For more information, see Section 14.5.1.

If you use currently use SYSMAN or SYSGEN to change parameters, and you have not added your customized parameter settings to MODPARAMS.DAT, follow the instructions in Section 14.3 to convert your customized parameter settings to MODPARAMS.DAT before running AUTOGEN.

14.3 Converting Your Customized Parameter Settings for Use with AUTOGEN

Digital recommends you use the AUTOGEN command procedure to tune your system. For more information on AUTOGEN, see Section 14.4. If you use the System Management utility (SYSMAN) or the System Generation utility (SYSGEN) to modify system parameter values, and you do not include these changes in the AUTOGEN parameter file MODPARAMS.DAT, these changes will be overridden the next time you run AUTOGEN.

If you used SYSMAN or SYSGEN to change parameter values in the past, use the following procedure to convert your parameter settings to work with AUTOGEN. This procedure explains how to add your customized parameter settings to MODPARAMS.DAT so they will be retained when you run AUTOGEN.

Before performing this task, you should understand AUTOGEN, feedback, and the AUTOGEN parameter file MODPARAMS.DAT, as explained in Section 14.4.

1. Save the parameter values that the system is now using as follows:

   ```
   $ RUN SYS$SYSTEM:SYSMAN
   SYSMAN> PARAMETERS USE ACTIVE
   SYSMAN> PARAMETERS WRITE SYS$SYSTEM:nodename_PARAMS_CURRENT.PAR
   ```

2. Write a listing of the active parameter values to an ASCII file named *nodename*_PARAMS.OLD as follows:

   ```
   SYSMAN> PARAMETERS SHOW/ALL/OUTPUT=nodename_PARAMS.OLD
   SYSMAN> PARAMETERS SHOW/SPECIAL/OUTPUT=nodename_PARAMS_SPECIAL.OLD
   SYSMAN> EXIT
   $ APPEND nodename_PARAMS_SPECIAL.OLD nodename_PARAMS.OLD
   ```

   You will use this file in step 6.

3. Edit AUTOGEN's parameter file SYS$SYSTEM:MODPARAMS.DAT to define symbols to specify values for the following:

   - Parameter values that are not calculated by AUTOGEN, such as SCSNODE and SCSSYSTEMID. See the AUTOGEN description in the *OpenVMS System Management Utilities Reference Manual* for a table of the parameters calculated by AUTOGEN.
   - Any parameter values that need to be adjusted to suit your system work load, for example, GBLPAGES and GBLSECTIONS.

   To specify a value, define symbols using the format MIN_parameter, MAX_parameter, or ADD_parameter rather than specifying an explicit value. For example:

   ```
   $ EDIT SYS$SYSTEM:MODPARAMS.DAT

   SCSNODE = "MYNODE"  ! Not calculated by AUTOGEN
   SCSSYSTEMID = 10001  ! Not calculated by AUTOGEN
   MIN_GBLPAGES = 10000  ! Needed for MCS, BLISS32, and ADA
   MIN_GBLSECTIONS = 600  ! Needed for MCS, BLISS32, and ADA
   ```

   To help you track the changes you make in MODPARAMS.DAT, add comments to each line, preceded by the comment character (!). For information on defining symbols in MODPARAMS.DAT, see Section 14.5.1.

4. Run AUTOGEN, but do *not* reboot. Use one of the following commands, depending on your system:

   - If the system has run a typical work load for more than 24 hours since last booting:

     ```
     $ @SYS$UPDATE:AUTOGEN SAVPARAMS SETPARAMS FEEDBACK
     ```

     The SAVPARAMS phase collects feedback information about resource use on the running system; this information is used by AUTOGEN. This command creates a feedback report named SYS$SYSTEM:AGEN$PARAMS.REPORT, which tells you about peak resource use.

   - If you want to use a previously collected feedback file:

     ```
     $ @SYS$UPDATE:AUTOGEN GETDATA SETPARAMS FEEDBACK
     ```

     If you start from the GETDATA phase, AUTOGEN does not collect current feedback.

   - If this is a new system (that is, it has no feedback) or the system has had little activity since last boot (for example, over the weekend) so there is no valid feedback file:

     ```
     $@SYS$UPDATE:AUTOGEN GETDATA SETPARAMS CHECK_FEEDBACK
     ```

     Use CHECK_FEEDBACK to let AUTOGEN determine whether the feedback is valid.

5. Write a listing of the new parameter values to an ASCII file as follows:

   ```
   $ RUN SYS$SYSTEM:SYSMAN
   SYSMAN> PARAMETERS USE CURRENT
   SYSMAN> PARAMETERS SHOW /ALL /OUTPUT=nodename_PARAMS.NEW;
   SYSMAN> PARAMETERS SHOW /SPECIAL /OUTPUT=nodename_PARAMS_SPECIAL.NEW;
   SYSMAN> EXIT
   $ APPEND nodename_PARAMS_SPECIAL.NEW; nodename_PARAMS.NEW;
   ```

6. Compare the old and new parameter values as follows:

   ```
   $ DIFFERENCES/PARALLEL/OUTPUT=nodename_PARAMS.DIFF/MATCH=5  -
   _$ nodename_PARAMS.OLD nodename_PARAMS.NEW
   ```

7. Print the differences file you created in step 6 (named in the format *nodename*_PARAMS.DIFF). Print the file on a 132-column line printer to make the output easier to read.

8. Compare the numbers in the two columns following each of the parameter name columns. The left column shows the old value; the right column shows the new value. The following figure illustrates sample output:

Parameter Names

	Old Values			
GBLPAGES	77500	10000	512	–
SYSMWCNT	2400	500	40	1638
INTSTKPAGES	4	4	1	–
BALSETCNT	250	16	4	819
IRPCOUNT	8000	60	0	13516
IRPCOUNTV	32000	250	0	13516
WSMAX	32800	1024	60	20000
NPAGEDYN	1944576	360000	16384	–
NPAGEVIR	777328	1000000	16384	–
PAGEDYN	1516032	190000	10240	–
VIRTUALPAGECNT	150000	9216	512	100000
⋮	⋮			

	New Values			
GBLPAGES	81800	10000	512	–
SYSMWCNT	2800	500	40	1638
INTSTKPAGES	4	4	1	–
BALSETCNT	250	16	4	819
IRPCOUNT	7200	60	0	13516
IRPCOUNTV	28800	250	0	13516
WSMAX	65500	1024	60	20000
NPAGEDYN	3000000	360000	16384	–
NPAGEVIR	12000000	1000000	16384	–
PAGEDYN	1780056	190000	10240	–
VIRTUALPAGECNT	270144	9216	512	100000
⋮	⋮			

ZK-5175A-GE

9. Make any needed adjustments in MODPARAMS.DAT using symbols prefixed by MIN_, MAX_, or ADD_. For example, if AUTOGEN calculated a smaller value for GBLPAGES, you might specify a minimum value for this parameter as follows:

   ```
   MIN_GBLPAGES = 10000
   ```

   If you originally specified a parameter value in MODPARAMS.DAT (in step 3) but the parameter has not been changed, verify the following:

   - The parameter name is spelled correctly. In MODPARAMS.DAT, AUTOGEN sees parameter names as symbol assignments. AUTOGEN cannot equate a symbol to the corresponding system parameter unless it is spelled correctly. Look in AGEN$FEEDBACK.REPORT for any error messages AUTOGEN might have written.
   - The value is correct: count the digits and make sure there are no commas.
   - The parameter occurs only once in MODPARAMS.DAT.
   - The parameter is not commented out.

   For most parameters, if the new value is greater than the old value, you can accept AUTOGEN's setting. If the new value is less than the old value, Digital recommends that you retain the old value because the system may not have been using that resource when running AUTOGEN.

   For example, you might have used SYSMAN to increase GBLPAGES to 10,000 to accommodate layered products, but have not specified that change in MODPARAMS.DAT. AUTOGEN might calculate that the system needs only 5000 global pages. When you reboot after running AUTOGEN, not all of your layered products may be installed, and you might receive the system message GPTFULL, global page table full, indicating that the system needs more GBLPAGES.

10. Repeat from Step 3 until you are satisfied with the new parameter values.

    If needed, make further changes in MODPARAMS.DAT, run AUTOGEN again, and verify the changes as before. Usually after this second pass of AUTOGEN, the parameter values will be stable and you can then reboot.

11. Reboot. When you reboot, the system will use the new parameter values. You do not need to use AUTOGEN to reboot, and you do not need to reboot right away. You do need to reboot before the new parameter values will be used by the system.

If the system does not boot, perform a conversational boot and use the backup parameter file you created in step 1:

```
SYSBOOT> USE SYS$SYSTEM:nodename_PARAMS_CURRENT.PAR
SYSBOOT> CONTINUE
```

When you enter the CONTINUE command, the system boots with the parameter values you saved before running AUTOGEN.

After the system has booted, if you need to return to the old parameter values you can enter the following commands:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETERS USE SYS$SYSTEM:nodename_PARAMS_CURRENT.PAR
SYSMAN> PARAMETERS WRITE CURRENT
SYSMAN> EXIT
```

12. Run AUTOGEN using feedback regularly to ensure that the resources of your system match your system work load. For information about running AUTOGEN using feedback, see Section 14.5.

14.4 Understanding the AUTOGEN Command Procedure

The AUTOGEN command procedure, SYS$UPDATE:AUTOGEN.COM, is provided on your distribution kit, and runs automatically when your system is installed or upgraded to set appropriate values for system parameters. In addition, Digital recommends you run AUTOGEN when you want to reset values for system parameters or to resize page, swap, and dump files. The new values and file sizes take effect the next time the system boots.

AUTOGEN only calculates certain significant system parameters. See the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual* for a table of system parameters affected by AUTOGEN calculation.

When to Run AUTOGEN

Digital recommends running AUTOGEN in the following circumstances:

- During a new installation or upgrade. (This happens automatically as part of the installation or upgrade procedure.)
- Whenever your work load changes significantly.
- When you add an optional (layered) software product. See the specific product documentation for installation requirements. Certain layered products might require you to execute AUTOGEN to adjust parameter values and page and swap file sizes. (For information on using AUTOGEN to modify page and swap files, see Section 15.14.)
- When you install images with the /SHARED attribute; the GBLSECTIONS and GBLPAGES parameters might need to be increased to accommodate the additional global pages and global sections consumed.
- On a regular basis to monitor changes in your system's work load. You can automate AUTOGEN to regularly check feedback and recommend system parameter changes. Section 14.6 describes a batch-oriented command procedure that runs AUTOGEN in feedback mode on a regular basis and automatically sends the feedback report to an appropriate MAIL account.

AUTOGEN Operations

AUTOGEN executes in phases. Depending on which phases you direct AUTOGEN to execute, it performs some or all of the following operations:

- Collects the following types of data:
  - Feedback (from the running system)
  - The hardware configuration (from the system)
  - Parameter requirements supplied by you
  - Parameter requirements supplied by Digital
- Calculates appropriate new values for significant system parameters (listed in the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual*).
- Creates a new installed image list.
- Calculates the sizes of system page, swap, and dump files.
- Adjusts the sizes of system page, swap, and dump files values of system parameter values, if necessary. (AUTOGEN invokes the System Management utility [SYSMAN] to change parameter values.)
- Optionally shuts down and reboots the system.

Invoking AUTOGEN

To invoke AUTOGEN, enter a command in the following format at the DCL prompt:

```
@SYS$UPDATE:AUTOGEN [start-phase] [end-phase] [execution-mode]
```

Where:

start-phase	Is the phase where AUTOGEN is to begin executing. Section 14.4.3 lists the AUTOGEN phases.
end-phase	Is the phase where AUTOGEN is to complete executing. Section 14.4.3 lists the AUTOGEN phases.
execution-mode	Is one of the following: • FEEDBACK—Use feedback. • NOFEEDBACK—Do not use feedback. • CHECK_FEEDBACK—Use feedback if it is valid. If feedback is invalid, ignore it, but continue executing through the end phase. • Blank (no execution mode specified)—Use feedback if it is valid. If it is not valid, quit before making any modifications.

For detailed information about invoking AUTOGEN, and the command line parameters you can specify, see the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual*.

Controlling AUTOGEN Operations

Table 14–1 summarizes the methods for controlling AUTOGEN behavior.

Table 14–1 Controlling AUTOGEN

To Control...	Use This Method...
Which operations AUTOGEN is to perform	Specify a start phase and an end phase when you invoke AUTOGEN. For detailed information about AUTOGEN phases, see the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual.*
Parameter values set by AUTOGEN	Specify values in the AUTOGEN parameter file MODPARAMS.DAT. You should periodically examine the results of calculations that AUTOGEN makes to determine whether AUTOGEN has drawn the correct conclusions about your hardware configuration and to be sure the system parameter values are appropriate for your workload requirements. If the values are not appropriate, adjust them by specifying desired values in MODPARAMS.DAT. For more information on MODPARAMS.DAT, see Section 14.4.4.
AUTOGEN's use of feedback information	Specify an execution mode when you invoke AUTOGEN. AUTOGEN can often improve system performance by using dynamic feedback gathered from the running system. However, feedback information is not always valid or appropriate. For more information, see Section 14.4.1.

14.4.1 AUTOGEN Feedback

AUTOGEN feedback minimizes the need for you to modify parameter values or system file sizes. Instead, feedback allows AUTOGEN to automatically size the operating system based on your actual work load. **Sizing** is the process of matching the allocation of system resources (memory and disk space) with the workload requirements of your site.

Feedback is information, continuously collected by the operating system executive, about the amount of various resources the system uses to process its work load. The information is collected when exception events occur, so the collection does not affect system performance. When run in **feedback mode**, AUTOGEN analyzes this information and adjusts any related parameter values.

AUTOGEN feedback affects the following resources (for a complete list of the affected system parameters, see the AUTOGEN section of the *OpenVMS System Manager's Manual*):

- Nonpaged pool
- Paged pool
- Lock resources
- Number of processes
- Global pages
- Global sections
- File system caches
- Page files

- Swap files

Feedback is gathered during AUTOGEN's SAVPARAMS phase and is written to the file SYS$SYSTEM:AGEN$FEEDBACK.DAT. This file is then read during the GETDATA phase. (See Section 14.4.3 for more information on AUTOGEN phases.)

Feedback is useful only if it accurately reflects the system's normal work load. For this reason, AUTOGEN performs some basic checks on the feedback and issues a warning message for either of the following conditions:

- The system has been up for less than 24 hours.
- The feedback is over 30 days old.

Whenever you modify the system (for example, a hardware upgrade, a change in the number of users, an optional product installation), you should operate in the new system environment for a period of time, and then execute AUTOGEN again starting from the SAVPARAMS phase.

VAX

On VAX systems, you can define the logical name AGEN$FEEDBACK_REQ_TIME to specify, in hours, a minimum age required for feedback. For more information, see Section 14.5.2. ♦

When AUTOGEN runs, it displays whether feedback is used, as follows:

```
Feedback information was collected on 21-JAN-1994 14:00:08.53
Old values below are the parameter values at the time of collection.
The feedback data is based on 21 hours of up time.
Feedback information will be used in the subsequent calculations
```

See the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual* for a table of the system parameters affected by AUTOGEN feedback,

14.4.2 The Feedback Report (AGEN$PARAMS.REPORT)

You must decide if you want to use the system parameter values and system file sizes calculated by AUTOGEN. To help in your decision making, AUTOGEN generates a report file (SYS$SYSTEM:AGEN$PARAMS.REPORT) that includes the following information:

- All parameters and system files directly affected by the feedback
- Current values
- New values
- The feedback used in each parameter calculation
- Any user- or Digital-supplied modifications found in MODPARAMS.DAT
- Any advisory or warning messages displayed during AUTOGEN's operations
- VAX On VAX systems, any user- or Digital-supplied modifications found in VMSPARAMS.DAT ♦
- AXP On AXP systems, the parameters found during the GENPARAMS phase. ♦

Example 14–1 shows the contents of a sample AUTOGEN feedback report for a VAX system. On AXP systems, the feedback report is similar, but not identical, to this example.

Example 14–1 Sample AUTOGEN Feedback Report

```
AUTOGEN Parameter Calculation Report on node: NODE22
  This information was generated at 23-JUN-1994 01:45:47.87
  AUTOGEN was run from GETDATA to TESTFILES using FEEDBACK

** No changes will be done by AUTOGEN **
   The values given in this report are what AUTOGEN would
    have set the parameters to.

Processing Parameter Data files
-------------------------------

** WARNING ** - The system was up for less than 24 hours when the feedback
information was recorded. This could result in feedback information
that does not accurately reflect your typical work load.

Including parameters from: SYS$SYSTEM:MODPARAMS.DAT

The following was detected within MODPARAMS.DAT
   Please review immediately.

** INFORMATIONAL ** - Multiple MIN values found for MIN_CHANNELCNT.
 Using MODPARAMS value (550) which is superseding OpenVMS value (255)

** INFORMATIONAL ** - Multiple MIN values found for MIN_SWPOUTPGCNT.
 Using MODPARAMS value (1000) which is superseding OpenVMS value (500)

** INFORMATIONAL ** - Multiple MIN values found for MIN_PQL_DWSEXTENT.
 Using MODPARAMS value (11000) which is superseding OpenVMS value (1024)

** INFORMATIONAL ** - Multiple MIN values found for MIN_PQL_MWSEXTENT.
 Using MODPARAMS value (11000) which is superseding OpenVMS value (1024)

Feedback information was collected on 22-JUN-1994 14:00:07.70
  Old values below are the parameter values at the time of collection.
  The feedback data is based on 13 hours of up time.
  Feedback information will be used in the subsequent calculations


Parameter information follows:
------------------------------

MAXPROCESSCNT parameter information:
 Feedback information.
    Old value was 100, New value is 80
    Maximum Observed Processes: 52

Information on VMS executable image Processing:

 Processing SYS$MANAGER:VMS$IMAGES_MASTER.DAT


GBLPAGFIL parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 1024.  The new value is 6024.
    GBLPAGFIL has been increased by 5000.
    GBLPAGFIL is not allowed to be less than 6024.


GBLPAGES parameter information:
 Feedback information.
    Old value was 43300, New value is 50000
    Peak used GBLPAGES: 36622
    Global buffer requirements: 6024
```

(continued on next page)

Example 14–1 (Cont.) Sample AUTOGEN Feedback Report

```
GBLSECTIONS parameter information:
 Feedback information.
    Old value was 400, New value is 400
    Peak used GBLSECTIONS: 294
 Override Information - parameter calculation has been overridden.
    The calculated value was 350.  The new value is 400.
    GBLSECTIONS is not allowed to be less than 400.

LOCKIDTBL parameter information:
 Feedback information.
    Old value was 2943, New value is 3071
    Current number of locks: 1853
    Peak number of locks: 3200

LOCKIDTBL_MAX parameter information:
 Feedback information.
    Old value was 65535, New value is 65535

RESHASHTBL parameter information:
 Feedback information.
    Old value was 1024, New value is 1024
    Current number of resources: 957

MSCP_LOAD parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 1.  The new value is 0.
    MSCP_LOAD has been disabled by a hard-coded value of 0.

MSCP_BUFFER parameter information:
 Feedback information.
    Old value was 128, New value is 128
    MSCP server I/O rate: 0 I/Os per 10 sec.
    I/Os that waited for buffer space: 0
    I/Os that fragmented into multiple transfers: 0

SCSCONNCNT parameter information:
 Feedback information.
    Old value was 5, New value is 5
    Peak number of nodes: 1
    Number of CDT allocation failures: 0

SCSRESPCNT parameter information:
 Feedback information.
    Old value was 300, New value is 300
    RDT stall count: 0

SCSBUFFCNT parameter information:
 Feedback information.
    Old value was 512, New value is 512
    CIBDT stall count: 0


NPAGEDYN parameter information:
 Feedback information.
    Old value was 686592, New value is 783360
    Maximum observed non-paged pool size: 815616 bytes.
    Non-paged pool request rate: 47 requests per 10 sec.


LNMSHASHTBL parameter information:
 Feedback information.
    Old value was 1024, New value is 1024
    Current number of shareable logical names: 1194
```

(continued on next page)

Example 14–1 (Cont.) Sample AUTOGEN Feedback Report

```
ACP_DIRCACHE parameter information:
 Feedback information.
    Old value was 88, New value is 88
    Hit percentage: 99%
    Attempt rate: 0 attempts per 10 sec.

ACP_DINDXCACHE parameter information:
 Feedback information.
    Old value was 25, New value is 25
    Hit percentage: 97%
    Attempt rate: 1 attempts per 10 sec.

ACP_HDRCACHE parameter information:
 Feedback information.
    Old value was 88, New value is 106
    Hit percentage: 98%
    Attempt rate: 17 attempts per 10 sec.

ACP_MAPCACHE parameter information:
 Feedback information.
    Old value was 8, New value is 8
    Hit percentage: 2%
    Attempt rate: 4 attempts per 10 sec.

PAGEDYN parameter information:
 Feedback information.
    Old value was 521728, New value is 542208
    Current paged pool usage: 304160 bytes.
    Paged pool request rate: 1 requests per 10 sec.

PFRATL parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 0.  The new value is 1.
    PFRATL has been disabled by a hard-coded value of 1.

WSDEC parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 35.  The new value is 19.
    WSDEC has been disabled by a hard-coded value of 19.

MPW_LOLIMIT parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 120.  The new value is 2100.
    MPW_LOLIMIT is not allowed to be less than 2100.

MPW_HILIMIT parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 1310.  The new value is 4500.
    MPW_HILIMIT is not allowed to be less than 4500.

LONGWAIT parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 30.  The new value is 10.
    LONGWAIT has been disabled by a hard-coded value of 10.

WSMAX parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 8200.  The new value is 12000.
    WSMAX is not allowed to be less than 12000.
```

(continued on next page)

Example 14–1 (Cont.) Sample AUTOGEN Feedback Report

```
PROCSECTCNT parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 32.  The new value is 40.
    PROCSECTCNT is not allowed to be less than 40.

PQL_DWSEXTENT parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 400.  The new value is 11000.
    PQL_DWSEXTENT is not allowed to be less than 11000.

PQL_MWSEXTENT parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 2048.  The new value is 11000.
    PQL_MWSEXTENT is not allowed to be less than 11000.

VAXCLUSTER parameter information:
 Override Information - parameter calculation has been overridden.
    The calculated value was 1.  The new value is 0.
    VAXCLUSTER has been disabled by a hard-coded value of 0.

Page, Swap, and Dump file calculations

  Page and Swap file calculations.

PAGEFILE1_SIZE parameter information:
 Feedback information.
    Old value was 45200, New value is 50500
    Maximum observed usage: 25265
 PAGEFILE1_SIZE will be modified to hold 50500 blocks

PAGEFILE2_SIZE parameter information:
 Feedback information.
    Old value was 154000, New value is 194400
    Maximum observed usage: 97175
 PAGEFILE2_SIZE will be modified to hold 194400 blocks

 ** WARNING ** - The disk on which PAGEFILE2 resides would be
    over 95% full if it were modified to hold 194400 blocks.
    NODE22$DKA300:[SYSTEM_FILES]PAGEFILE.SYS will not be modified.
    NODE22$DKA300:[SYSTEM_FILES]PAGEFILE.SYS will remain at 154002
blocks.

SWAPFILE1_SIZE parameter information:
 Feedback information.
    Old value was 15000, New value is 15000
    Maximum observed usage: 14280
 Override Information - parameter calculation has been overridden.
    The calculated value was 21400.  The new value is 15000.
    SWAPFILE1_SIZE is not allowed to exceed 15000.
 SWAPFILE1 will not be modified.

SWAPFILE2_SIZE parameter information:
 Feedback information.
    Old value was 50000, New value is 26300
    Maximum observed usage: 1680
 SWAPFILE2_SIZE will be modified to hold 26300 blocks

 ** WARNING ** - The disk on which SWAPFILE2 resides would be
    over 95% full if it were modified to hold 26300 blocks.
    NODE22$DKA300:[SYSTEM_FILES]SWAPFILE.SYS will not be modified.
    NODE22$DKA300:[SYSTEM_FILES]SWAPFILE.SYS will remain at 50001 blocks.
```

(continued on next page)

Example 14–1 (Cont.) Sample AUTOGEN Feedback Report

```
 Dumpfile calculations:

No dump file modifications would have been made.
Dumpfile will remain at 34116 blocks.
```

14.4.3 AUTOGEN Phases

When you invoke AUTOGEN, you specify a start phase and an end phase for AUTOGEN to execute. AUTOGEN executes all phases from the start phase to the end phase. Depending on the start phase and end phase you specify, AUTOGEN can execute any of the following phases, in the order shown in Table 14–2.

Table 14–2 AUTOGEN Phases

Phase	Description
SAVPARAMS	Saves dynamic feedback from the running system.
GETDATA	Collects all data to be used in AUTOGEN calculations.
GENPARAMS	Generates new system parameters; creates the installed image list.
TESTFILES	Displays the system page, swap, and dump file sizes calculated by AUTOGEN (cannot be used as a start phase).
GENFILES	Generates new system page, swap, and dump files if appropriate (cannot be used as a start phase).
SETPARAMS	Runs SYSMAN to set the new system parameters in the default parameter file, saves the original parameters, and generates a new parameter file, AUTOGEN.PAR. On VAX systems, the default parameter file is VAXVMSSYS.PAR. The original parameters are saved in the file VAXVMSSYS.OLD. On AXP systems, the default parameter file is ALPHAVMSSYS.PAR. The original parameters are saved in the file ALPHAVMSSYS.OLD.
SHUTDOWN	Prepares the system to await a manual reboot.
REBOOT	Automatically shuts down and reboots the system.
HELP	Displays help information to the screen.

For detailed information about each AUTOGEN phase and the files affected by each phase, see the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual.*

14.4.4 The AUTOGEN Parameter File (MODPARAMS.DAT)

AUTOGEN reads a parameter file named MODPARAMS.DAT during the GETDATA phase. You can add commands to this file to control the system parameter values and file sizes that AUTOGEN sets. You can use MODPARAMS.DAT to do the following:

Operation	For More Information
Increase the value of any numeric system parameter	Section 14.5.1.1
Set a minimum value for a numeric system parameter	Section 14.5.1.2
Set a maximum value for a numeric system parameter	Section 14.5.1.3
Specify an absolute value for a system parameter	Section 14.5.1.4
Include an external parameter file	Section 14.5.3
Specify sizes of page, swap, and dump files	Section 15.14.1.1
†Define the number of VAXcluster nodes	Section 14.5.1.5
†Define the number of Ethernet adapters	Section 14.5.1.6
†Preset parameter values before adding memory	Section 14.5.1.7
Specify an alternate default startup command procedure	Section 4.4.2

†VAX specific

To help track changes you make to MODPARAMS.DAT, make sure you add comments, preceded by the comment character (!), each time you change the file.

Caution

The recommended method of changing system parameters and system file sizes is to edit MODPARAMS.DAT to include parameter settings. If you change a system parameter value or file size using SYSMAN, SYSGEN, or a conversational boot, and you do not specify the value in MODPARAMS.DAT, AUTOGEN will recalculate the value or file size the next time it runs. For more information, see Section 14.5.1.

Example

The following example shows the contents of a sample MODPARAMS.DAT file:

```
!
! ***************** A Sample MODPARAMS.DAT for Node NODE22 ***************
!
! MODPARAMS.DAT for "NODE22"
! REVISED: 09/13/94 -CHG- Upped GBLPAGES to account for ADA.
!
SCSNODE          = "NODE22"      ! This is not calculated by AUTOGEN.
SCSSYSTEMID      = 19577         ! This is not calculated by AUTOGEN.
TTY_DEFCHAR2     = %X0D34        ! This is not calculated by AUTOGEN.
ADD_ACP_DIRCACHE= 150            ! Hit rate was only 65% on directory cache.
MIN_PAGEDYN      = 500000        ! PAGEDYN must be at least 1/2 Mbyte to
                                 ! account for a large number of logical names.
!
MAX_PAGEFILE1_SIZE  = 15000      ! Maximum size for primary page.
MAX_SWAPFILE        =  5000      ! Maximum size for swap file space.
MAX_DUMPFILE        = 32768      ! Maximum size for dump file space.

ADD_GBLPAGES     = 425+507+157   ! Account for MCS, BLISS32 and ADA.
ADD_GBLSECTIONS = 4 + 5 + 2      ! Account for MCS, BLISS32 and ADA.
VIRTUALPAGECNT   = 144264        ! So that we can read MONSTR's 68Mb dumps.
!
! end of MODPARAMS.DAT for NODE22
```

14.5 Modifying System Parameters with AUTOGEN

The recommended method of modifying system parameters is to execute AUTOGEN in two passes, as follows:

1. First pass—Execute AUTOGEN using the following command:

   ```
   $ @SYS$UPDATE:AUTOGEN SAVPARAMS GENPARAMS
   ```

   This command instructs AUTOGEN to do the following:

   - Save the current feedback
   - Gather all of the information required for the calculations
   - Calculate the system parameter values
   - Generate the feedback report
   - Write the information to SETPARAMS.DAT

   Review the input to the calculations (PARAMS.DAT), the output from the calculations (SETPARAMS.DAT), and the report generated (AGEN$PARAMS.REPORT).

   If you are not satisfied with the parameter settings, modify parameter values by editing MODPARAMS.DAT as explained in Section 14.5.1. Then reexecute AUTOGEN from the GETDATA phase.

   When you are satisfied with the contents of SETPARAMS.DAT, go on to step 2.

2. Second pass—Execute AUTOGEN a second time using the following command:

   ```
   $ @SYS$UPDATE:AUTOGEN GENPARAMS REBOOT
   ```

   This AUTOGEN command runs SYSMAN to update the new system parameter values and installs them on the system when it is rebooted.

14.5.1 Controlling AUTOGEN's Parameter Settings with MODPARAMS.DAT

If, after examining the AGEN$PARAMS.REPORT or SETPARAMS.DAT file, you decide to correct hardware configuration data or modify system parameter values chosen by AUTOGEN, edit the MODPARAMS.DAT file as described in this section to manually specify parameter values.

Caution

Always edit MODPARAMS.DAT to specify values for parameters. Do not edit PARAMS.DAT; modifying the contents of this file might prevent AUTOGEN from operating correctly.

For information on editing MODPARAMS.DAT to control sizes of page, swap, and dump files, see Section 15.14.1.1.

You can define symbols in MODPARAMS.DAT using the following formats to control parameter values:

Control Method	Symbol Format	For More Information
Increment a value by a specified amount	ADD_*	Section 14.5.1.1
Specify a minimum value	MIN_*	Section 14.5.1.2
Specify a maximum value	MAX_*	Section 14.5.1.3
Specify an absolute value	Parameter name	Section 14.5.1.4

When defining symbols in MODPARAMS.DAT, make sure of the following:

- The value is correct and valid for the parameter. Count the digits. Do not use commas.
- The symbol occurs only once in MODPARAMS.DAT.
- The symbol value is not commented out.

Caution

When AUTOGEN reads MODPARAMS.DAT or any other parameter file, it checks to determine if the symbol names specified in the file are valid. If they are not, AUTOGEN writes a warning message to AGEN$PARAMS.REPORT. However, AUTOGEN checks only the symbol name; it does not check the validity of the value specified for the symbol.

If a value is invalid, the line is *not* ignored. AUTOGEN attempts to use the specified value.

A symbol is not checked if it is specified in a line that contains a DCL expression other than the symbol assignment (=). For example, AUTOGEN does not check the validity of a symbol name specified in a line with the DCL IF statement. Instead, AUTOGEN writes a warning message to AGEN$PARAMS.REPORT.

To help track changes you make to MODPARAMS.DAT, make sure you add comments preceded by the comment character (!) each time you change the file.

14.5.1.1 Incrementing a Value with the ADD_ Prefix

Use the ADD_ prefix to increment the value of any NUMERIC parameter. The new values are updated in subsequent AUTOGEN calculations during the GENPARAMS phase. The following example demonstrates the use of the ADD_ prefix:

```
ADD_GBLPAGES=500
ADD_NPAGEDYN=10000
```

An ADD_ parameter record for a parameter that AUTOGEN calculates will add the value toAUTOGEN's calculations. An ADD_ parameter record for a parameter that AUTOGEN does not calculate will add the value to the parameter's default (not current) value. (See the AUTOGEN section on the *OpenVMS System Management Utilities Reference Manual* for a table of parameters affected by AUTOGEN.)

Note

The ADD_ value is added to the calculated value once, and does not accumulate with successive runs for feedback calculations.

Typically, you would not use the ADD_ prefix for modifying parameters that are calculated by the feedback mechanism, because the feedback results should accurately reflect your work load. However, if you do use the ADD_ prefix with feedback, be aware that AUTOGEN will add a value only once if AUTOGEN is run to the SETPARAMS phase or beyond. If you wish to maintain a minimum level above AUTOGEN's calculation, use the MIN_ prefix.

14.5.1.2 Specifying a Minimum Value with the MIN_ Prefix

Use the MIN_ prefix if you do not want AUTOGEN to set a parameter below a specified value. MIN_ refers to the minimum value to which a parameter can be set by AUTOGEN.

```
MIN_PAGEDYN = 400000
```

AXP On AXP systems, AUTOGEN does not reduce system parameter values that allocate resources. In effect, it considers current values to be base values. ♦

14.5.1.3 Specifying a Maximum Value with the MAX_ Prefix

Use the MAX_ prefix if you do not want AUTOGEN to set a parameter above a specified value. MAX_ refers to the maximum value to which a parameter can be set by AUTOGEN.

```
MAX_PAGEDYN = 400000
```

14.5.1.4 Specifying an Absolute Value

Use this method to specify a value for a parameter that AUTOGEN does not calculate. (See the AUTOGEN section of the *OpenVMS System Management Utilities Reference Manual* for a table of the system parameters modified in AUTOGEN calculations.)

Note

Digital strongly recommends that you use this method only for parameters that describe the system environment (for example, SCSNODE and SCSSYSTEMID). For the parameters that AUTOGEN calculates, specifying a value with this method disables AUTOGEN's calculations. Instead of specifying an absolute value, use one of the following methods:

- Specify a minimum value with the MIN_ prefix (see Section 14.5.1.2)
- Specify a maximum value with the MAX_ prefix (see Section 14.5.1.3)
- Increment the value with the ADD_ prefix (see Section 14.5.1.1)

To specify an absolute parameter value, add an assignment statement in the following format to MODPARAMS.DAT:

parameter = parameter-value ! comment

For example, the following command assigns the node name BIGVAX to the SCSNODE parameter:

```
SCSNODE = "BIGVAX"   ! the node name
```

14.5.1.5 Defining the Number of VAXcluster Nodes (VAX Only)

VAX

On VAX systems in a VAXcluster environment, use the NUM_NODES symbol to prevent temporary changes in VAXcluster membership from affecting AUTOGEN's calculation of VAXcluster-related parameter values. Define the NUM_NODES symbol in MODPARAMS.DAT to specify the number of nodes that are to run in the VAXcluster. AUTOGEN uses this value to set parameters that are affected by the number of VAXcluster nodes.

For example, you might include the following line in MODPARAMS.DAT:

```
NUM_NODES = 30 ♦
```

14.5.1.6 Defining the Number of Ethernet Adapters (VAX Only)

VAX

On VAX systems, in a VAXcluster environment, use the NUM_ETHERADAPT symbol to prevent temporary changes in VAXcluster membership from affecting AUTOGEN's calculation of VAXcluster-related parameter values. Define the NUM_ETHERADAPT symbol in MODPARAMS.DAT to specify the total number of Ethernet adapters in the VAXcluster.

For example, you might include the following line in MODPARAMS.DAT:

```
NUM_ETHERADAPT = 40 ♦
```

14.5.1.7 Presetting Parameter Values Before Adding Memory (VAX Only)

VAX

On VAX systems, if you are planning to upgrade your system hardware by adding a large amount (512 MB or more) of memory, you might want to preset your system parameters to values appropriate for the additional memory. Presetting your system parameters minimizes the possibility of memory upgrade problems caused by inappropriate parameter values.

How to Perform This Task

Perform the following steps:

1. Add a line in the following format to SYS$SYSTEM:MODPARAMS.DAT:

   MEMSIZE = total-number-of-pages-of-memory-after-upgrade.

   For example:

   ```
   MEMSIZE = 2048 * 1024 ! (2048 page per MB *  1GB of memory)
   ```

2. Run AUTOGEN to the SETPARAMS phase.
3. Perform the hardware upgrade to add the additional memory.
4. Edit MODPARAMS.DAT to remove the line added in step 1. ♦

14.5.2 Specifying a Minimum Required Age for Feedback (VAX Only)

VAX

On VAX systems, AUTOGEN feedback is useful only when a system has been running long enough to accurately reflect the system's normal work load. By default, AUTOGEN uses feedback if the data is older than 24 hours. On VAX systems, you can define the logical name AGEN$FEEDBACK_REQ_TIME to specify, in hours, a different minimum age required for feedback. AUTOGEN uses this value to determine whether the feedback is to be used.

For example, you might define the logical name as follows, to indicate that AUTOGEN should use feedback if it is older than 19 hours:

```
$ DEFINE/SYSTEM AGEN$FEEDBACK_REQ_TIME 19
```

To define this logical name each time the system starts up, add this command to SYLOGICALS.COM. ♦

14.5.3 Including an External Parameter File in MODPARAMS.DAT

You can include external parameter files in MODPARAMS.DAT. For example, you might want to set a system parameter to the same value on all nodes in a VAXcluster or VMScluster; you might also want to specify node-specific values for other system parameters. You could specify the cluster-common values in a separate cluster-common file and include this cluster-common file in the MODPARAMS.DAT file on each system in the VAXcluster or VMScluster.

To include a parameter file, place a command in the following format in MODPARAMS.DAT, or in any parameter file that is included in MODPARAMS.DAT:

AGEN$INCLUDE_PARAMS full-directory-spec:filename

Example

To include a cluster-common parameter file named CLUSTERPARAMS.DAT, create a common parameter file with the following name:

SYS$COMMON:[SYSEXE]CLUSTERPARAMS.DAT

Add the following line in the MODPARAMS.DAT file in the system-specific directory of each VMScluster system:

```
AGEN$INCLUDE_PARAMS SYS$COMMON:[SYSEXE]CLUSTERPARAMS.DAT
```

14.6 Automating AUTOGEN Reports

Digital recommends you create a batch-oriented command procedure to automatically run AUTOGEN on a regular basis and send the resulting feedback reports to an appropriate MAIL account. Example 14–2 provides a sample command procedure.

Note

This command procedure runs AUTOGEN only to recommend system parameter values and send you a report. It does not run AUTOGEN to change system parameters or reboot the system. If, after reviewing the report, you decide to change system parameters, follow the instructions in Section 14.6.1.

The command procedure in Example 14–2 runs two passes of AUTOGEN. On the first pass, AUTOGEN runs during peak workload times to collect data on realistic system work loads. This pass does not degrade system performance. On the second pass, AUTOGEN runs during off-peak hours to interpret the data collected in the first stage.

The procedure sends the resulting report, contained in the file AGEN$PARAMS.REPORT, to the SYSTEM account. Review this report on a regular basis to see whether the load on the system has changed.

Example 14–2 shows a sample command procedure. Use this procedure only as an example; create a similar command procedure as necessary to meet the needs of your configuration.

Example 14–2 Sample AUTOGEN Command Procedure

```
$ BEGIN$:   ! ++++++++++ AGEN_BATCH.COM ++++++++++
$  on warning then goto error$
$  on error then goto error$
$  on severe_error then goto error$
$  on control_y then goto error$
$!
$! Setup process
$!
$! Set process information
$  set process/priv=all/name="AUTOGEN Batch"
$! Keep log files to a reasonable amount
$  purge/keep=5 AGEN_Batch.log
$  time = f$time()    ! Fetch current time
$  hour = f$integer(f$cvtime(time,,"hour")) ! Get hour
$  today = f$cvtime(time,,"WEEKDAY")  ! Get Day of the week
$  if f$integer(f$cvtime(time,,"minute")) .ge. 30 then hour = hour + 1
$!
$! Start of working day...
$!
$ 1AM$:
$  if hour .le. 2
$     then
$     next_time = "today+0-14"
$     gosub submit$    ! Resubmit yourself
$     set noon
$!

$!      Run AUTOGEN to TESTFILES using the parameter values collected earlier

$!      in the day (i.e., yesterday at 2:00pm)
$       if today .eqs. "Tuesday" .OR. today .eqs. "Thursday" .OR. -
    today .eqs. "Saturday"
$          then
$      @sys$update:autogen GETDATA TESTFILES feedback ❷
$      mail/sub="AUTOGEN Feedback Report for system-name" -
  sys$system:agen$params.report system ❸
$      ! Clean up
$            purge/keep=7 sys$system:agen$feedback.report ❹
$            purge/keep=7 sys$system:agen$feedback.dat
$            purge/keep=7 sys$system:params.dat
$            purge/keep=7 sys$system:autogen.par
$            purge/keep=7 sys$system:setparams.dat
$            purge/keep=7 sys$system:agen$addhistory.tmp
$            purge/keep=7 sys$system:agen$addhistory.dat
$        endif
$     goto end$
$     endif
```

(continued on next page)

Example 14–2 (Cont.) Sample AUTOGEN Command Procedure

```
$!
$ 2PM$:
$  if hour .le. 15
$     then
$     next_time = "today+0-17"
$     gosub submit$
$     if today .eqs. "Monday" .OR. today .eqs. "Wednesday" .OR. -
  today .eqs. "Friday"
$        then
$          @sys$update:autogen SAVPARAMS SAVPARAMS feedback ❶
$        endif
$     goto end$
$     endif
$!
$ 5PM$:
$  if hour .le. 18
$     then
$     next_time = "tomorrow+0-1"
$     gosub submit$
$     endif
$!
$! End of working day...
$!
$ END$:     ! ---------- BATCH.COM ----------
$  exit
$!++
$! Subroutines
$!--
$!
$ SUBMIT$:
$  submit/name="AGEN_Batch"/restart/noprint - ❺
   /log=AGEN_batch.log -
   /queue=sys$batch/after="''next_time'" sys$system:AGEN_batch.com
$  return
$!++
$! Error handler
$!--
$ ERROR$:
$  mail/sub="AGEN_BATCH.COM - Procedure failed." _nl: system
$  goto end$
```

The commands in this procedure perform the following tasks:

❶ Executes the first pass of AUTOGEN during peak workload times to collect data on realistic work loads. This command runs a very fast image so it does not degrade system response.

❷ Executes the second pass of AUTOGEN during off-peak hours to interpret the data collected in the first pass.

❸ Mails the resulting report file named AGEN$PARAMS.REPORT to the SYSTEM account.

❹ Cleans up the files created.

❺ Resubmits the command procedure.

14.6.1 Changing Parameter Values After Reviewing AUTOGEN Reports

If the command procedure report described in the previous section shows AUTOGEN's calculations are different from the current values, correct the tuning by executing AUTOGEN with one of the two following commands:

- If the system can be shut down and rebooted immediately, execute the following command:

  ```
  $ @SYS$UPDATE:AUTOGEN GETDATA REBOOT FEEDBACK
  ```

- If the system cannot be shut down and rebooted immediately, execute the following command to reset the system parameters:

  ```
  $ @SYS$UPDATE:AUTOGEN GETDATA SETPARAMS FEEDBACK
  ```

  The new parameters will take effect the next time the system boots.

14.7 Managing System Parameters with the System Management Utility (SYSMAN)

Note

Digital recommends you use AUTOGEN to modify system parameters. For more information, see Section 14.5. There may be times, however, when you want to view system parameters for a group of nodes or change parameters temporarily. SYSMAN enables you to do this.

The System Management utility (SYSMAN) provides the ability to inspect and modify system parameters for an entire VMScluster or for any group of nodes, rather than just one system. The PARAMETERS commands available in SYSMAN duplicate the parameter functions of the OpenVMS System Generation utility (SYSGEN).

You can use SYSMAN to manage system parameters as follows:

Task	For More Information
Show parameter values	Section 14.7.2
Modify current values in the parameter file	Section 14.7.3
Modify active values on a running system[1]	Section 14.7.4

[1]Applies only to the dynamic system parameters.

SYSMAN provides the commands and functions shown in Table 14–3.

Table 14–3 SYSMAN PARAMETERS Commands

Command	Function
PARAMETERS SHOW	Displays parameter values.
PARAMETERS USE	Reads a set of parameters from memory or disk into the work area for inspection or modification.

(continued on next page)

Table 14–3 (Cont.) SYSMAN PARAMETERS Commands

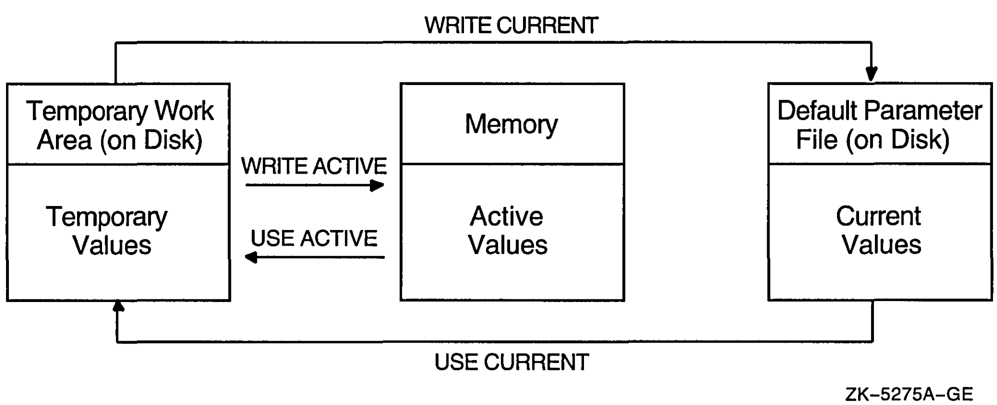
Command	Function
PARAMETERS SET	Changes parameter values only in the work area; more permanent modification requires the PARAMETERS WRITE command.
PARAMETERS WRITE	Writes the content of the work area to memory or to disk.

For more information about the temporary work area, see the next section.

14.7.1 Understanding Parameter Values and SYSMAN

You should understand the different system parameter values explained in Section 14.1.1. Briefly, **current values** are values stored in the default parameter file on disk. **Active values** are values that are stored in memory and used while the system is running. In addition to these values, SYSMAN writes a temporary copy into its own work area on disk. Figure 14–1 illustrates these different sets of values and how SYSMAN commands affect them.

Figure 14–1 SYSMAN Temporary, Active, and Current Parameters



In a typical session, you might display and change values in the following sequence:

1. Read values into SYSMAN's temporary work space with the PARAMETERS USE command.

   PARAMETERS USE ACTIVE reads in active values.

   PARAMETERS USE CURRENT reads in current values.

2. Display the parameter values with the PARAMETERS SHOW command.
3. Change a value with the PARAMETERS SET command. You must use the PARAMETERS WRITE command to activate the value.
4. Make the change effective with the PARAMETERS WRITE command.

   PARAMETERS WRITE ACTIVE writes the value to the set of active values. (You can change an active value only if the parameter is a dynamic parameter.) PARAMETERS WRITE CURRENT writes the value to the set of current values.

For a list of all the system parameters, see the *OpenVMS System Management Utilities Reference Manual*.

14.7.2 Showing Parameter Values with SYSMAN

You can use the SYSMAN command PARAMETERS SHOW to display parameter values for all the nodes in a cluster.

Examples

1. The following example shows one method to display information about parameters. In this case, using the /LGI qualifier displays all login security control parameters. You can display many categories of parameters, such as /ACP, /ALL, and /SPECIAL. See the *OpenVMS System Management Utilities Reference Manual* for a complete list of parameters and parameter categories.

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> PARAMETERS SHOW/LGI

Parameters in use: Active
Parameter Name          Current    Default    Min.     Max.     Unit  Dynamic
--------------          -------    -------   -------  -------   ----  -------
LGI_BRK_TERM                  0          1         0        1 Boolean    D
LGI_BRK_DISUSER               0          0         0        1 Boolean    D
LGI_PWD_TMO                  30         30         0      255 Seconds    D
LGI_RETRY_LIM                 3          3         0      255 Tries      D
LGI_RETRY_TMO                20         20         0      255 Seconds    D
LGI_BRK_LIM                   5          5         0      255 Failures   D
LGI_BRK_TMO                 300        300         0       -1 Seconds    D
LGI_HID_TIM                 300        300         0       -1 Seconds    D
```

2. This example invokes SYSMAN and specifies the environment to be the local cluster, which consists of NODE21 and NODE22. The example also displays the active value for the LGI_BRK_TMO parameter, which controls the number of seconds that a user, terminal, or node is permitted to attempt login. In this case, it is 600.

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, Current command environment:
   Clusterwide on local cluster
   Username MORIN    will be used on nonlocal nodes
SYSMAN> PARAMETERS SHOW LGI_BRK_TMO

Node NODE21:   Parameters in use: ACTIVE
Parameter Name         Current  Default  Minimum    Maximum Unit  Dynamic
--------------         -------  -------  -------    ------- ----  -------
LGI_BRK_TMO                600      300        0         -1 Seconds     D

Node NODE22:   Parameters in use: ACTIVE
Parameter Name         Current  Default  Minimum    Maximum Unit  Dynamic
--------------         -------  -------  -------    ------- ----  -------
LGI_BRK_TMO                600      300        0         -1 Seconds     D
```

14.7.3 Modifying a Parameter File with SYSMAN

You can use the SYSMAN command PARAMETERS WRITE to write system parameter values and the name of the site-independent startup command procedure to your choice of parameter file or the current system parameter file on disk.

The PARAMETERS WRITE CURRENT command sends a message to OPCOM to record the event, unless you have changed the system message format with the DCL command SET MESSAGE.

Note

The PARAMETERS WRITE CURRENT command writes *all* of the active or current parameter values—not just the one you may be working on—to disk.

Examples

1. The following example creates a new parameter specification file:

```
SYSMAN> PARAMETERS WRITE SYS$SYSTEM:NEWPARAM
```

2. When used with the PARAMETERS SET command, the PARAMETERS WRITE command modifies the current system parameter file on disk:

```
SYSMAN> PARAMETERS SET LGI_BRK_TMO 300
SYSMAN> PARAMETERS WRITE CURRENT
```

14.7.4 Modifying Active Values with SYSMAN

Using the SYSMAN commands PARAMETERS SET, PARAMETERS WRITE, and PARAMETERS USE enables you to modify active parameter values.

Modifying active values immediately affects dynamic parameters by changing their values in memory. The *OpenVMS System Management Utilities Reference Manual* identifies the dynamic parameters, as does the SYSMAN command PARAMETERS SHOW/DYNAMIC. Values for nondynamic parameters cannot be changed while the system is running.

Modifying active values does not affect current values in the system parameter file on disk, because the next time you boot the system, the values on disk are established as the active values.

If you set new active parameter values and you want to use the new values for subsequent boot operations, write the new values to the current parameter file with the PARAMETERS WRITE CURRENT command, as shown in the Examples section.

Caution

Parameter values modified with SYSMAN will be overridden by the AUTOGEN command procedure. To keep parameter modifications made with SYSMAN, edit the file SYS$SYSTEM:MODPARAMS.DAT as explained in Section 14.5.1 to specify the new parameter values.

Examples

1. The following example changes the LGI_BRK_TMO value to 300 in the work area, writes this change into memory as an active value, and displays the active value:

```
SYSMAN> PARAMETERS SET LGI_BRK_TMO 300

SYSMAN> PARAMETERS WRITE ACTIVE
SYSMAN> PARAMETERS SHOW LGI_BRK_TMO
```

```
Node NODE21:   Parameters in use: ACTIVE
Parameters in use: ACTIVE
Parameter Name         Current   Default   Minimum    Maximum Unit  Dynamic
--------------         -------   -------   -------    ------- ----  -------
LGI_BRK_TMO                300       300         0         -1 Seconds     D

Node NODE22:   Parameters in use: ACTIVE
Parameter Name         Current   Default   Minimum    Maximum Unit  Dynamic
--------------         -------   -------   -------    ------- ----  -------
LGI_BRK_TMO                300       300         0         -1 Seconds     D
```

2. The following example calls the current parameter values, including LGI_BRK_TMO, from disk to the work area, then displays LGI_BRK_TMO. In this example, the current value on disk is 600.

```
SYSMAN> PARAMETERS USE CURRENT
SYSMAN> PARAMETERS SHOW LGI_BRK_TMO

Node NODE21:   Parameters in use: CURRENT
Parameter Name         Current   Default   Minimum    Maximum Unit  Dynamic
--------------         -------   -------   -------    ------- ----  -------
LGI_BRK_TMO                600       300         0         -1 Seconds     D

Node NODE22:   Parameters in use: CURRENT
Parameter Name         Current   Default   Minimum    Maximum Unit  Dynamic
--------------         -------   -------   -------    ------- ----  -------
LGI_BRK_TMO                600       300         0         -1 Seconds     D
```

3. The next example writes the LGI_BRK_TMO value of 600 from the work area to memory, where it becomes the active value on the running system. Note that the command PARAMETER WRITE ACTIVE writes all the parameter values from the work area into memory, not just the value of LGI_BRK_TMO.

```
SYSMAN> PARAMETERS WRITE ACTIVE

SYSMAN> PARAMETERS USE ACTIVE

SYSMAN> PARAMETERS SHOW LGI_BRK_TMO

Node NODE21:   Parameters in use: ACTIVE
Parameter Name         Current   Default   Minimum    Maximum Unit  Dynamic
--------------         -------   -------   -------    ------- ----  -------
LGI_BRK_TMO                600       300         0         -1 Seconds     D

Node NODE22:   Parameters in use: ACTIVE
Parameter Name         Current   Default   Minimum    Maximum Unit  Dynamic
--------------         -------   -------   -------    ------- ----  -------
LGI_BRK_TMO                600       300         0         -1 Seconds     D
```

14.8 Managing System Parameters with the System Generation Utility (SYSGEN)

Note

Digital recommends you use AUTOGEN to modify system parameters. For more information, see Section 14.5. If for some reason you cannot use AUTOGEN, Digital recommends you use the System Management utility (SYSMAN). For more information, see Section 14.7.

Although it is not the recommended method, you can also use the System Generation utility (SYSGEN) to manage system parameters as follows:

Task	For More Information
Show parameter values	Section 14.8.2
Modify current values in the default parameter file	Section 14.8.3
Modify active values on a running system[1]	Section 14.8.4
Create a new parameter file	Section 14.8.5

[1]Applies only to the dynamic system parameters.

SYSGEN provides the commands shown in Table 14–4 for managing system parameters. See the SYSGEN section of the *OpenVMS System Management Utilities Reference Manual* for detailed descriptions of SYSGEN commands.

Table 14–4 SYSGEN Commands Used with System Parameters

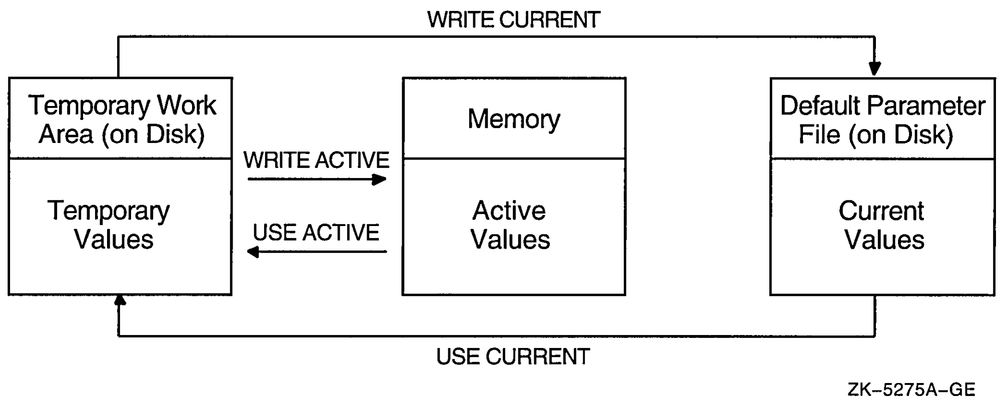
Command	Function
SHOW	Displays parameter values.
USE	Reads a set of values from memory or disk into a temporary work area for inspection or modification.
SET	Changes parameter values only in the work area; more permanent modification requires the WRITE command.
WRITE	Writes the content of the work area to memory or to disk.

For more information about the temporary work area, see the next section.

14.8.1 Understanding Parameter Values and SYSGEN

You should understand the different system parameter values explained in Section 14.1.1. Briefly, **current values** are values stored in the default parameter file on disk. **Active values** are values that are stored in memory and used while the system is running. In addition to these values, SYSGEN writes a temporary copy into its own work area on disk. Figure 14–2 illustrates these different sets of values and shows how SYSGEN commands affect them.

Figure 14–2 SYSGEN Temporary, Active, and Current Parameter Values



In a typical session, you might display and change values in the following sequence:

1. Read values into SYSGEN's temporary work space with the USE command. USE ACTIVE reads in active values. USE CURRENT reads in current values.
2. Display the parameter values with the SHOW command.
3. Change a value with the SET command. (Note, however that the SET command only changes the value in SYSGEN's temporary work area.)
4. Make the change effective with the WRITE command.

   WRITE ACTIVE writes the value to the set of active values in memory. (You can change an active value only if the parameter is a dynamic parameter.) WRITE CURRENT writes the value to the set of current values on disk.

For a list of all the system parameters, see the *OpenVMS System Management Utilities Reference Manual*.

14.8.2 Showing Parameter Values with SYSGEN

To display values for system parameters, perform the following steps:

1. Invoke SYSGEN by entering the following command:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   ```

2. Enter the USE command to specify which values you want to display, as follows:

To Display	Enter
Active values	USE ACTIVE
Current values	USE CURRENT
Values from another parameter file	USE *file-spec* For *file-spec*, specify the parameter file from which you want to display values; for example, USE SYS$SYSTEM:ALTPARAMS.DAT

3. Enter a SHOW command in the following format:

   SHOW [/qualifier] [parameter-name]

   Specify qualifiers to display parameters grouped by type. For example:

To Display Values For	Enter
The WSMAX parameter	SHOW WSMAX
All dynamic parameters	SHOW/DYNAMIC
All parameters in the TTY category	SHOW/TTY
All parameters	SHOW/ALL

For more information on the SYSGEN SHOW command and qualifiers, see the SYSGEN section of the *OpenVMS System Management Utilities Reference Manual*.

Example

The following example uses SYSGEN to show the current values of all TTY system parameters:

```
$ RUN SYS$SYSTEM:SYSGEN
$ USE CURRENT
SYSGEN> SHOW/TTY

Parameters in use: Current❶
Parameter Name            Current   Default      Min.      Max. Unit       Dynamic
--------------            -------   -------   -------   ------- ----       -------
❷                              ❸         ❹         ❺         ❻ ❼
TTY_SCANDELTA            10000000  10000000    100000        -1 100Ns
TTY_DIALTYPE                    0         0         0       255 Bit-Encode
TTY_SPEED                      15        15         1        16 Special
TTY_RSPEED                      0         0         0        16 Special
TTY_PARITY                     24        24         0       255 Special
TTY_BUF                        80        80         0     65535 Characters
TTY_DEFCHAR             402657952 402657952         0        -1 Bit-Encode
TTY_DEFCHAR2               135178      4098         0        -1 Bit-Encode
TTY_TYPAHDSZ                   78        78         0        -1 Bytes
TTY_ALTYPAHD                 2048       200         0     32767 Bytes
TTY_ALTALARM                  750        64         0        -1 Bytes
TTY_DMASIZE                    64        64         0        -1 Bytes       D ❽
TTY_PROT                    65520     65520         0        -1 Protection
TTY_OWNER                   65540     65540         0        -1 UIC
TTY_CLASSNAME               "TTY"     "TTY"      "AA"      "ZZ" Ascii
TTY_SILOTIME                    8         8         0       255 Ms
TTY_TIMEOUT                  3600       900         0        -1 Seconds    D
TTY_AUTOCHAR                    7         7         0       255 Character  D
SYSGEN>
```

SYSGEN displays the following information:

❶ The values in use (in this example, current values)

❷ The name of the system parameter

❸ The value requested (in this example, the current value). The heading of this column is always "Current," regardless of whether it displays the current or active value of the parameter. In this context, "Current" refers to the value of this parameter *currently* in use, as specified by the USE command; it does not refer to the *current value* of the parameter stored on disk with the WRITE CURRENT command.

❹ The default value

❺ The minimum value

❻ The maximum value

❼ The unit of allocation

❽ A "D," if the system parameter is dynamic

14.8.3 Modifying the System Parameter File with SYSGEN

Caution

Parameter values modified with the System Generation utility (SYSGEN) will be overridden by the AUTOGEN command procedure. To keep parameter modifications made with SYSGEN, edit the file SYS$SYSTEM:MODPARAMS.DAT as explained in Section 14.5.1 to specify the new parameter values.

Note

Although you can modify system parameter values with SYSGEN, Digital recommends you use AUTOGEN. For more information, see Section 14.5.

If you cannot use AUTOGEN, Digital recommends you use the System Management utility (SYSMAN) to modify system parameters. For more information, see Section 14.7.

Modifying the current values in the default system parameter file has no immediate effect on active values on a running system. However, during subsequent boot operations, the system is initialized with the new values.

Example

The following example modifies the TTY_TIMEOUT parameter value in the VAX system parameter file:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> USE CURRENT
SYSGEN> SET TTY_TIMEOUT 3600
SYSGEN> WRITE CURRENT
%OPCOM, 15-APR-1994 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process
ID 00160030 into file VAXVMSSYS.PAR
SYSGEN> EXIT
```

14.8.4 Modifying Active Values with SYSGEN

Caution

Parameter values modified with SYSGEN will be overridden by the AUTOGEN command procedure. To keep parameter modifications made with SYSGEN, edit the file SYS$SYSTEM:MODPARAMS.DAT as explained in Section 14.5.1 to specify the new parameter value.

Note

Although you can modify system parameter values with SYSGEN, Digital recommends you use AUTOGEN or the System Management utility (SYSMAN). For more information, see Section 14.7.

Modifying active values immediately affects dynamic parameters by changing their values in memory. The *OpenVMS System Management Utilities Reference Manual* identifies the dynamic parameters (as does the SYSGEN command SHOW/DYNAMIC). Values for nondynamic parameters cannot be changed while the system is running.

Modifying active values does not affect the current values in the system parameter file on disk. The next time you boot the system, the old current values are established as the active values.

If you set new active parameter values (by entering WRITE ACTIVE) and you want to use the new values for subsequent boot operations, you must write the new values to the current parameter file on disk by entering the WRITE CURRENT command, as explained in Section 14.8.3. If the parameters are not dynamic parameters, you must enter the WRITE CURRENT command and reboot the system.

When you change active parameters with SYSGEN, the Operator Communication Manager (OPCOM) writes a message to the operator log and the operator console, unless you have changed the system message format with the DCL command SET MESSAGE.

Examples

1. The following example modifies the active value of the PFCDEFAULT parameter:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> SET PFCDEFAULT 127
SYSGEN> WRITE ACTIVE
%OPCOM, 15-APR-1994 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITEACT, ACTIVE system parameters modified by process
ID 00160030
SYSGEN> EXIT
```

2. The following example modifies the active value of the PFCDEFAULT parameter and also writes it to the AXP system parameter file, so it will be used when the system reboots:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> SET PFCDEFAULT 127
SYSGEN> WRITE ACTIVE
%OPCOM, 15-APR-1994 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITEACT, ACTIVE system parameters modified by process
ID 00160030
SYSGEN> WRITE CURRENT
%OPCOM, 15-APR-1994 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process
ID 00160030 into file ALPHAVMSSYS.PAR
SYSGEN> EXIT
```

14.8.5 Creating a New Parameter File with SYSGEN

Creating a new parameter file has no effect on the running system. During a subsequent conversational boot operation, however, you can initialize the active system with the values of the new file.

How to Perform This Task

1. Invoke SYSGEN by entering the following commands:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
```

2. Enter a command in the following format to write a copy of a parameter file into SYSGEN's temporary workspace:

   USE file-spec

   Where *file-spec* is the file specification for the parameter file to be used as a base. You will modify the values in this file to create a new parameter file.

3. Enter commands in the following form to modify values as needed:

   SET parameter-name parameter-value

   For *parameter-name*, specify the name of the parameter to be changed. For *parameter-value*, specify the new value.

4. Specify a command in the following format to write the values to a new parameter file:

   WRITE file-spec

   where *file-spec* is the file specification for the parameter file to be created.

5. Exit SYSGEN.

Caution

Parameter values modified with the System Generation utility (SYSGEN) will be overridden by the AUTOGEN command procedure. To keep parameter modifications made with SYSGEN, edit the file SYS$SYSTEM:MODPARAMS.DAT as explained in Section 14.5.1 to specify the new parameter values.

Examples

1. The following example creates a new version of the parameter file PARAMS.PAR with a new value for the TTY_TIMEOUT parameter:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> USE SYS$MANAGER:PARAMS.PAR
SYSGEN> SET TTY_TIMEOUT 3600
SYSGEN> WRITE SYS$MANAGER:PARAMS.PAR
SYSGEN> EXIT
```

2. The following example creates a file named SYS$SYSTEM:OURSITE.PAR, using the PARAMS.PAR file as a base:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN> USE SYS$MANAGER:PARAMS.PAR
SYSGEN> SET TTY_TIMEOUT 1000
SYSGEN> WRITE OURSITE.PAR
SYSGEN> EXIT
```

14.9 Modifying System Parameters with a Conversational Boot

Note

Although you can modify system parameters with a conversational boot, Digital recommends you use AUTOGEN or the System Management utility (SYSMAN). For more information, see Section 14.5 and Section 14.7.

Use a conversational boot only to change isolated system parameters *temporarily* or in an emergency. For example, during a system upgrade, you would use a conversational boot to modify STARTUP_P1 to use a minimum startup.

Remember that if you change a value and do not add the changed value to the AUTOGEN parameter file MODPARAMS.DAT, AUTOGEN will overwrite the value the next time AUTOGEN executes.

With a conversational boot operation, you can modify the active parameter values in the following ways before the system boots:

Task	For More Information
Modify active values for individual parameters	Section 4.2.1
Initialize active values using values stored in a parameter file other than the default parameter file	Section 4.2.2
Reinitialize active values using default values	Section 4.3.1

At the end of the conversational boot, the default system parameter file is modified to store the new active parameter values.

Caution

Parameter values modified with a conversational boot will be overridden by the AUTOGEN command procedure. To keep parameter modifications made with a conversational boot, edit the file SYS$SYSTEM:MODPARAMS.DAT as explained in Section 14.5.1 to specify the new parameter values.

15

Managing System Page, Swap, and Dump Files

The system page, swap, and dump files are created by default. However, you should understand these files. In addition, you might want to change them to meet the needs of your site.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Displaying information about page and swap files	Section 15.3
Manually calculating an appropriate size for the system page, swap, and dump files	Section 15.4
Minimizing dump file size when disk space is insufficient	Section 15.5
Using SDA to analyze the contents of a crash dump	Section 15.6
†Using the CLUE utility to obtain historical information about crash dumps	Section 15.7
Copying dump files to tape or disk	Section 15.8
Saving the contents of the system dump file after a system failure	Section 15.9
Freeing dump information from the page file	Section 15.10
Creating page and swap files	Section 15.11
Installing page and swap files	Section 15.12
Removing page, swap, and dump files	Section 15.13
Changing page, swap, and dump file sizes	Section 15.14
Controlling page, swap, and dump file sizes in MODPARAMS.DAT	Section 15.14.1.1

†VAX specific

This chapter explains the following concepts:

Concept	Section
The system dump file	Section 15.1
Page and swap files	Section 15.2
†CLUE	Section 15.7.1

†VAX specific

15.1 Understanding the System Dump File

When the operating system detects an unrecoverable error or an inconsistency within itself that causes the system to fail, it writes the contents of the error log buffers, processor registers, and memory into the **system dump file**, overwriting its previous contents.

When writing the system dump file, the system displays a number of console messages and information about the error or inconsistency. The following message tells you that the dump file was successfully written:

```
System dump complete
```

Caution

Be sure to wait until the system dump file is complete and you see this message before using the console terminal to halt the system. If you don't, your system might not save a complete dump file.

The contents of the console messages and the contents of the system dump file are important sources of information in determining the cause of a system failure. You use the contents in the following ways:

- Use the System Dump Analyzer utility (SDA) to analyze the contents of the dump and determine the cause of a failure.
- VAX On VAX systems, use CLUE to obtain historical information from system dump files. ♦
- Send the contents of the dump to Digital Equipment Corporation, along with a Software Performance Report (SPR).

The default system dump file, SYS$SPECIFIC:[SYSEXE]SYSDUMP.DMP, is furnished as an empty file in the operating system distribution kit.

AUTOGEN automatically determines an appropriate size for the system dump file for your hardware configuration and system parameters. For special configurations or varying work loads you might want to change the size of the system dump file. For information, see Section 15.14.1.

You do not need a system dump file to run the operating system. However, you need a system dump file to diagnose system crashes.

Using the Page File to Store System Crash Dumps

The operating system uses the latest version of SYS$SYSTEM:SYSDUMP.DMP to store system crash dumps. If SYSDUMP.DMP does not exist in SYS$SYSTEM, the operating system uses the system paging file, SYS$SYSTEM:PAGEFILE.SYS, overwriting the contents of that file. If the SAVEDUMP system parameter is set, the crash dump is retained in PAGEFILE.SYS when the system is booted. If SAVEDUMP is clear, the system uses the paging file for paging and any dump written to the paging file is lost.

If you use SYS$SYSTEM:PAGEFILE.SYS to capture system crash dumps, you should later free the space occupied by the dump for use in system paging, with either of the following methods:

- Use the SDA COPY command to copy the page file to a different file.

- Use the SDA RELEASE command to delete the information from the page file.

For detailed instructions, see Section 15.10.

Include the appropriate SDA command in the SYSTARTUP_VMS.COM startup command procedure to free dump information from the page file each time the system reboots.

Caution

Be careful when using the page file for selective dumps. Selective dumps use up all available space. If your page file is small, selective dump information might fill the entire page file, leaving no space for paging during system boot. This can cause the system to hang during reboot.

Types of Dumps

There are two types of dumps: physical and selective. Table 15–1 defines physical and selective dumps. Table 15–3 compares the information available in physical and selective dump files.

Table 15–1 Comparison of Physical and Selective Dumps

Type	Description
Physical dump	Writes the entire contents of physical memory to the dump file. To ensure a useful physical dump, the dump file must be large enough to contain all of physical memory.
Selective dump	Stores those portions of memory most likely to be useful in crash dump analysis. A selective dump is useful when disk space is not available to hold all of physical memory. To direct your system to save a selective dump, set the system parameter DUMPSTYLE to the appropriate value. For more information, see Section 15.5.

Requirements for Creating a Useful System Dump

The following requirements must be met for the operating system to write a useful system dump file:

- The system parameter DUMPBUG must be set to 1 (the default value).
- If the system parameter SAVEDUMP is set to 0 (the default) the file SYS$SPECIFIC:[SYSEXE]SYSDUMP.DMP must exist on the system disk.
- If the file SYS$SPECIFIC:[SYSEXE]SYSDUMP.DMP does not exist on the system disk, the page file must be used to store the dump. The system parameter SAVEDUMP must be set to 1 and the file SYS$SPECIFIC:[SYSEXE]PAGEFILE.SYS must exist on the system disk.
- If sufficient disk space is not available to allow a system dump file that can hold all of memory, the system parameter DUMPSTYLE must be set to the appropriate value to store a selective dump. For more information, see Section 15.5.
- The size of the system dump file (or page file if the SAVEDUMP system parameter is set) must be large enough to hold all information that is to be written if the system fails.

If the system parameter DUMPBUG is set, AUTOGEN automatically sizes SYSDUMP.DMP if enough disk space is available.

If the system parameter SAVEDUMP is set, AUTOGEN performs no operations on the dump file.

AUTOGEN sizes the page file only for paging use, regardless of whether the SAVEDUMP system parameter is set.

BACKUP Considerations

System dump files have the NOBACKUP attribute, so the Backup utility (BACKUP) does not copy them unless you use the qualifier /IGNORE=NOBACKUP when invoking BACKUP. When you use the SDA COPY command to copy the system dump file to another file, the operating system does not automatically set the new file to NOBACKUP. If you want to set the NOBACKUP attribute on the copy, use the SET FILE command with the /NOBACKUP qualifier as described in the *OpenVMS DCL Dictionary*.

Security Considerations

As included in the distribution kit, SYS$SYSTEM:SYSDUMP.DMP is protected against world access. Because a system dump file can contain privileged information, you should keep this level of protection on dump files. Similarly, when you copy dump files using the System Dump Analyzer utility (SDA) as explained in Section 15.9 and Section 15.10, be sure to protect the copy from world read access. For more information on file protection, see the *Security Guide*.

15.2 Understanding Page and Swap Files

As part of memory management, the operating system makes efficient use of physical memory by moving information between physical memory and files stored on disk. The system does this in two ways: **paging** and **swapping**. Table 15–2 defines these and related terms.

Table 15–2 Paging and Swapping Terminology

Term	Definition
Paging	To efficiently use the physical memory allotted to a *process*, the operating system moves infrequently used portions of a process workspace out of physical memory to a file. For more information on paging, see the *Guide to OpenVMS Performance Management*.
Page file	The file to which the system writes paged portions of memory. Your distribution kit includes a page file named SYS$SYSTEM:PAGEFILE.SYS. If necessary, SYS$SYSTEM:PAGEFILE.SYS can be used in place of the system crash dump file. For more information, see Section 15.1.
Swapping	To efficiently use the physical memory available for the *entire system*, the operating system moves the entire workspace of a less active process out of physical memory to a file. For more information on swapping, see the *Guide to OpenVMS Performance Management*.

(continued on next page)

Table 15–2 (Cont.) Paging and Swapping Terminology

Term	Definition
Swap file	The file to which the system writes swapped portions of memory. Your distribution kit includes a swap file named SYS$SYSTEM:SWAPFILE.SYS.
Primary page and swap files	The default page and swap files provided with your distribution kit. These files are named SYS$SYSTEM:PAGEFILE.SYS and SYS$SYSTEM:SWAPFILE.SYS.
Secondary page and swap files	Additional page and swap files that you might create for performance or disk space reasons. If you kept the primary page and swap file on the system disk, the system uses the space in the secondary files for paging and swapping in addition to the space in the primary page and swap files. For information on creating secondary page and swap files, see Section 15.11.

Installing Files

Page and swap files must be installed before the system can use them. The system automatically installs the latest versions of SYS$SYSTEM:PAGEFILE.SYS and SWAPFILE.SYS during startup. If you create secondary page and swap files, you must make sure the system installs them during startup. For more information on installing page and swap files, see Section 15.12.

File Sizes and Locations

AUTOGEN automatically determines appropriate sizes for the files for your hardware configuration and system parameters. For special configurations or varying work loads, you might want to change the size of the page or swap file. For information, see Section 15.14.1.

If your system does not require the page file for storing crash dumps, you can move it off the system disk. However, you should keep one page file on the system disk, if possible, so that you can boot the system if another disk holding the page files becomes unavailable. The swap file can also be moved off the system disk.

15.3 Displaying Information About Page and Swap Files

The DCL command SHOW MEMORY/FILES displays information about the page and swap files existing on your system, including file names, sizes, and the amount of space used. For example:

```
$ SHOW MEMORY/FILES
              System Memory Resources on 12-AUG-1994 11:54:20.06

Paging File Usage (pages):                     Free  Reservable      Total
  DISK$PAGE:[SYSEXE]SWAPFILE_IPL31.SYS;2      79992       79992      79992
  DISK$PAGE:[SYSEXE]PAGEFILE_IPL31.SYS;1      23263     -370027     249992
```

In the SHOW MEMORY/FILES display, concentrate on the columns labeled “Free” and “Total.” In general, the number in the “Free” column should be no less than half the number in the “Total” column.

Note that the number displayed in the column labeled “Reservable” can be a negative number. This number represents the amount of file space still available to be reserved by processes for paging. Processes can reserve more space than is available because it is unlikely that all the reserved space will be used for paging at one time.

15.4 Manually Calculating Appropriate Sizes for Dump, Page, and Swap Files

When you install or upgrade the operating system, AUTOGEN automatically calculates appropriate sizes for your system page, swap, and dump files based on your hardware configuration and system parameters. However, you might want to manually calculate the sizes for these files. The following sections describe how to determine appropriate sizes for the system page, swap, and dump files.

15.4.1 Calculating System Dump File Size

Sufficient space in the system dump file is critical to saving a complete crash dump. The AUTOGEN command procedure calculates an appropriate size for your dump file. However, if you want to manually calculate the dump file size, use one of the following formulas. These formulas calculate the size required to hold a physical dump.

For SYSDUMP.DMP

VAX

On VAX systems, use the following formula:

```
size-in-blocks(SYS$SYSTEM:SYSDUMP.DMP)
= size-in-pages(physical-memory)
+ (number-of-error-log-buffers * blocks-per-buffer)
+ 1 ♦
```

AXP

On AXP systems, use the following formula:

```
size-in-blocks(SYS$SYSTEM:SYSDUMP.DMP)
= size-in-pages (physical-memory)
* blocks-per-page
+ (number-of-error-log-buffers * blocks-per-buffer)
+ 2 ♦
```

where:

size-in-pages	Is the size of physical memory, in pages. Use the DCL command SHOW MEMORY to determine the total size of physical memory on your system.

blocks-per-page	Is the number of blocks per page of memory.
	On VAX systems, the disk block size and page size are identical (512).
	On AXP systems, calculate the number of blocks per page of memory by dividing the system's page size by 512 (the size of a block). Use the following commands:
	`$ PAGESIZE==F$GETSYI ("PAGE_SIZE")`
	`$ BLOCKSPERPAGE=PAGESIZE/512`
	`$ SHOW SYMBOL BLOCKSPERPAGE♦`

number-of-error-log-buffers	Is the value of the system parameter ERRORLOGBUFFERS. This parameter sets the number of error log buffers to permanently allocate in memory.
blocks-per-buffer	Is the value of the system parameter ERLBUFFERPAGES. This parameter sets the number of pages of memory in each buffer.

A large memory system or a system with small disk capacity may not be able to supply enough disk space for a full memory dump. Under these circumstances, you should set the system parameter DUMPSTYLE to the appropriate value to indicate that the system is to dump only selective information. For more information, see Section 15.5.

For PAGEFILE.SYS

If SYS$SYSTEM:SYSDUMP.DMP does not exist, the system writes crash dumps to the primary page file SYS$SYSTEM:PAGEFILE.SYS. The AUTOGEN command procedure calculates an appropriate size for your page file. However, if you want to manually calculate the minimum page file size required to hold crash dumps, use the following formula:

VAX

On VAX systems:

```
size-in-blocks (SYS$SYSTEM:PAGEFILE.SYS)
= size-in-pages (physical-memory)
+ (number-of-error-log-buffers * blocks-per-buffer)
+ 1
+ 1000 ♦
```

AXP

On AXP systems:

```
Size-in-blocks (SYS$SYSTEM:PAGEFILE.SYS)
= size-in-pages (physical-memory)
* blocks-per-page
+ (number-of-error-log-buffers * blocks-per-buffer)
+ 2
+ RSRVPAGCNT ♦
```

where:

size-in-pages	Is the size of physical memory, in pages. Use the DCL command SHOW MEMORY to determine the total size of physical memory on your system.

blocks-per-page	Is the number of blocks per page of memory.
	On VAX systems, the disk block size and page size are identical (512).
	On AXP systems, calculate the number of blocks per page of memory by dividing the system's page size by 512 (the size of a block). Use the following commands:
	`$ PAGESIZE==F$GETSYI ("PAGE_SIZE")`
	`$ BLOCKSPERPAGE=PAGESIZE/512`
	`$ SHOW SYMBOL BLOCKSPERPAGE`

number-of-error-log-buffers	Is the value of the system parameter ERRORLOGBUFFERS. This parameter sets the number of error log buffers to permanently allocate in memory.
blocks-per-buffer	Is the value of the system parameter ERLBUFFERPAGES. This parameter sets the number of pages of memory in each buffer.
RSRVPAGCNT	Is the value of the RSRVPAGCNT system parameter.

Caution

This formula calculates only the minimum size requirement for saving a dump in the system's primary page file. For most systems, the page file must be larger than this to avoid hanging the system. For more information about calculating the page file size, see Section 15.4.2.

15.4.2 Calculating Page File Size

Sufficient page file space is critical to system performance. The AUTOGEN command procedure calculates an appropriate size for your page file space. The size calculated by AUTOGEN should be sufficient. However, if you want to manually calculate the size for page file space, use one of the following formulas.

VAX

On VAX systems:

```
size-in-blocks (total for all page files on the system)
= size-of-average-program (in blocks)
* maximum-number-of-processes ♦
```

AXP

On AXP systems:

```
size-in-blocks (total for all page files on the system)
= size-of-average-program (in pagelets)
* maximum-number-of-processes ♦
```

where:

size-of-average-program	Is the value is the size of the average image running on the system. On VAX systems, specify this value in blocks. On AXP systems, specify this value in pagelets.
maximum-number-of-processes	Is the value of the MAXPROCESSCNT system parameter.

The size you calculate can be represented in one of the following ways:

- In the primary page file only
- Distributed across primary and secondary page files
- If you have removed the primary page file in SYS$SYSTEM, distributed across a number of secondary page files

Once you have determined an initial size for your page file or files (either with AUTOGEN, or manually), monitor page file usage by executing AUTOGEN with the following command:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES FEEDBACK
```

With this command, AUTOGEN writes page file usage and size recommendations to the feedback report AGEN$PARAMS.REPORT. (For more information on AUTOGEN and the feedback report, see Section 14.4 and Section 14.4.2.) The DCL command SHOW MEMORY/FILES also displays file usage, as explained in Section 15.2.

Keep page file usage less than half the size of the page file or files. If a paging space starts to fill to the point where system performance is being affected, a message will be printed on the console terminal. If this happens, increase the size of your page file or files or install additional files.

Note

Your system resources and work load affect the required size of your page file. You should be familiar with your system resources and work load. For more information, see the *Guide to OpenVMS Performance Management*.

You limit the amount of page file space consumed by user programs by using the /PGFLQUOTA qualifier of the AUTHORIZE commands ADD and MODIFY. (See the AUTHORIZE section in the *OpenVMS System Management Utilities Reference Manual* for more information.) You should not reduce the value of /PGFLQUOTA below 1024. Size requirements of the page file vary widely, depending on user applications.

15.4.3 Calculating Swap File Size

Sufficient swap file space is critical to system performance. The AUTOGEN command procedure calculates an appropriate size for your swap file space. If you want to manually calculate the size for swap file space, use the following formula:

```
size-in-blocks (total for all swap files on the system)
= Maximum-number-of-processes
* Average-working-set-quota-of-processes-on-system
```

where:

Maximum-number-of-processes	Is the value of the MAXPROCESSCNT system parameter.
Average-working-set-quota-of-processes-on-system	Is the average value of the WSQUOTA limit for processes running on the system. On VAX systems, specify the value in pages. On AXP systems, specify the value in pagelets.

On AXP and VAX systems, the size you calculate can be represented in any of the following ways:

- In the primary swap file only
- Distributed across primary and secondary swap files
- If you have removed the primary swap file in SYS$SYSTEM, distributed across a number of secondary swap files

Once you have determined an appropriate size for swapfile space (either manually or with AUTOGEN), monitor swap file usage with the DCL command SHOW MEMORY/FILES as explained in Section 15.3. Keep at least one-third of the swap file space unused; otherwise, system performance can be severely affected.

Note

Your system resources and work load determine the required size of your swap file. You should be familiar with your system resources and work load. For more information, see the *Guide to OpenVMS Performance Management*.

15.5 Minimizing Dump File Size When Disk Space Is Insufficient

In certain system configurations, it may be impossible to preserve the entire contents of memory in a disk file. For instance, a large memory system may not be able to supply enough disk space for a full memory dump. If your system attempts to save all of memory but the dump file is too small to accommodate the entire dump, the System Dump Analyzer utility (SDA) might not be able to analyze the dump.

VAX

On VAX systems, insufficient dump space would also prevent the Crash Logger Utility Extractor (CLUE) from being able to analyze the dump. ♦

On VAX and AXP systems, to preserve those portions of memory that contain information most useful in determining the causes of system failures, you can use selective dumps. Table 15–1 defines physical and selective dumps. Table 15–3 compares the information available in physical and selective dump files.

Table 15–3 Comparison of Physical and Selective Dump Files

Type	Available Information	Unavailable Information
Physical dump	Complete contents of physical memory in use, stored in order of increasing physical address and error log buffers.	Contents of paged-out memory at the time of the crash.
Selective dump	System page table, global page table, system space memory, error log buffers, and process and control regions (plus global pages) for all saved processes.	Contents of paged-out memory at the time of the crash, process and control regions of unsaved processes, and memory not mapped by a page table.

To direct your system to save selective dumps, set the system parameter DUMPSTYLE to the appropriate value. Table 15–4 defines possible values for the DUMPSTYLE parameter. For information on changing system parameter values, see Section 14.5.

VAX

On VAX systems, the default value of DUMPSTYLE is 0. ♦

AXP

On AXP systems, the default value of DUMPSTYLE is 1. ♦

Table 15–4 Possible Values for the DUMPSTYLE System Parameter

Value	Meaning
0	AUTOGEN attempts to create a dump file large enough to contain a physical dump (that is, all of physical memory). ‡ On AXP systems, the console output is much longer than on VAX systems. If the system crashes, this value provides shorter console output than the value of 2.
1	AUTOGEN attempts to create a dump file large enough to contain a selective dump (that is, only the information required for SDA to analyze a system failure). ‡On AXP systems, the console output is much longer than on VAX systems. If the system crashes, this value provides shorter console output than the value of 3.

‡AXP systems only

(continued on next page)

Table 15–4 (Cont.) Possible Values for the DUMPSTYLE System Parameter

Value	Meaning
‡2	On AXP systems, AUTOGEN attempts to create a dump file large enough to contain a physical dump (that is, all of physical memory). If the system crashes, it produces full console output.
‡3	On AXP systems, AUTOGEN attempts to create a dump file large enough to contain a selective dump (that is, only the information required for SDA to analyze a system failure). If the system crashes, it produces full console output.

‡AXP systems only

15.6 Using SDA to Analyze the Contents of a Crash Dump

The System Dump Analyzer utility (SDA) lets you interpret the contents of the dump file to investigate the probable causes of the crash. For information on analyzing a crash dump, see the *System Dump Analyzer Utility Manual.*

If your system fails, you should send Digital Equipment Corporation a Software Performance Report (SPR) and a copy of the system dump file written at the time of the failure. For information on copying the system dump file, see Section 15.8.

15.7 Using CLUE to Obtain Historical Information About Crash Dumps (VAX Only)

VAX

On VAX systems, the Crash Log Utility Extractor (CLUE) is a tool you can use to display the contents of a **crash history file**. By examining the contents of the crash history file, you might be able to understand and resolve the issues responsible for failures (crashes), and you might also obtain other useful data.

15.7.1 Understanding CLUE (VAX Only)

The crash history file, which is created and updated by CLUE, contains key parameters from crash dump files. Unlike crash dumps, which are overwritten with each system failure and are therefore typically available only for the most recent failure, the crash history file is a permanent record of system failures. CLUE is available on VAX systems.

After a system fails and physical memory is copied to the crash dump file, CLUE automatically appends the relevant parameters to the file CLUE$OUTPUT:CLUE$HISTORY.DATA when the system is restarted. The remainder of this section describes how you can use CLUE to display the data it has collected; reference information about CLUE is available in the *OpenVMS System Management Utilities Reference Manual.*

Note

The history file will typically grow by about 10-15 blocks for each entry. You can limit the number of entries in the binary file by defining the logical name CLUE$MAX_ENTRIES to be the maximum number desired. When this number is reached, the oldest entries are deleted from the history file.

By default, operator shutdowns are recorded in the history file. You can exclude information from operator shutdowns in the history file by defining the logical name CLUE$EXCLUDE_OPERS as being TRUE, for example by including the following line in SYS$MANAGER:SYSTARTUP_VMS.COM:

```
$ DEFINE /SYSTEM CLUE$EXCLUDE_OPERS TRUE
```

15.7.2 Displaying Data Using CLUE (VAX Only)

To display data using CLUE, you must first define the following symbol:

```
$ CLUE :== $CLUE
```

After defining the symbol, you can use CLUE to display information by entering the following command:

```
$ CLUE/DISPLAY
CLUE_DISPLAY>
```

At the CLUE_DISPLAY> prompt, you can issue commands to do the following:

- Use the DIRECTORY command to list failures that have occurred since a specified date, failures of a particular type, failures that contain a specified module, and failures that have a specified offset.

  For example, you can list all the failures in the history file using the DIRECTORY command, as follows:

  ```
  CLUE_DISPLAY> DIRECTORY
  ```

- Use the SHOW command to generate information similar to that obtained from certain commands in the System Dump Analyzer utility (SDA).

  For example, if you wanted complete information on the crash listed as crash number 7, the following SHOW command would provide the information:

  ```
  CLUE_DISPLAY> SHOW ALL 7
  ```

- Use the EXTRACT command to write the data from an entry to a file.

  For example, the following command writes the data from entry number 7 in the crash history file to a file named 15MAYCRASH.TXT:

  ```
  CLUE_DISPLAY> EXTRACT 7/OUTPUT=15MAYCRASH.TXT
  ```

For more information about CLUE commands, see the *OpenVMS System Management Utilities Reference Manual*. ♦

15.8 Copying Dump Files to Tape or Disk

If your system fails, you should send a copy of the contents of the system dump file to Digital Equipment Corporation along with a Software Performance Report (SPR). You can use the Backup utility (BACKUP) to create save sets containing system dump files on magnetic tape or disk. However, when using BACKUP to copy dump files, you must specify the /IGNORE=(NOBACKUP,NOINTERLOCK) qualifier for the following reasons:

- By default, the system dump file has the NOBACKUP attribute, so it is not copied unless you specify /IGNORE=NOBACKUP.
- The system keeps an open channel to the dump file, so the file is not copied unless you specify /IGNORE=INTERLOCK.

For more information on using BACKUP, see Section 10.13.2. For information on BACKUP commands, see the BACKUP section in the *OpenVMS System Management Utilities Reference Manual*.

15.9 Saving the Contents of the System Dump File After a System Failure

If the system fails, it overwrites the contents of the system crash dump file and the previous contents are lost. For this reason, you should ensure that your system automatically analyzes and copies the contents of the dump file each time the system reboots. To do so, modify the site-specific startup command procedure SYSTARTUP_VMS.COM so that it invokes the System Dump Analyzer utility (SDA) when the system is booted.

Be aware of the following information:

- When invoked from the site-specific startup procedure in the STARTUP process, SDA executes the specified commands only if the system is booting immediately after a system failure. If the system is rebooting after it was shut down with SHUTDOWN.COM or OPCCRASH.EXE, SDA exits without executing the commands.
- You can use the DCL COPY command to copy the dump file; however, the SDA COPY command is preferred because it marks the dump file as copied. This is particularly important if the dump was written into the paging file, SYS$SYSTEM:PAGEFILE.SYS, because it releases those pages occupied by the dump to the pager. For more information, see Section 15.10.
- Because a system dump file can contain privileged information, you should protect copies of dump files from world read access. For more information on file protection, see the *Security Guide*.
- System dump files have the NOBACKUP attribute, so the Backup utility (BACKUP) does not copy them unless you use the qualifier /IGNORE=NOBACKUP when invoking BACKUP. When you use the SDA COPY command to copy the system dumpfile to another file, the operating system does not automatically set the new file to NOBACKUP. If you want to set the NOBACKUP attribute on the copy, use the SET FILE command with the /NOBACKUP qualifier as described in the *OpenVMS DCL Dictionary*.

Example

The SDA COPY command in the following example saves the contents of the file SYS$SYSTEM:SYSDUMP.DMP and performs some analysis of the file:

```
$ !
$ !     Print dump listing if system just failed
$ !
$ ANALYZE/CRASH_DUMP SYS$SYSTEM:SYSDUMP.DMP
    COPY SYS$SYSTEM:SAVEDUMP.DMP        ! Save dump file
    SET OUTPUT DISK1:SYSDUMP.LIS        ! Create listing file
    READ/EXECUTIVE                      ! Read in symbols for kernel
    SHOW CRASH                          ! Display crash information
    SHOW STACK                          ! Show current stack
    SHOW SUMMARY                        ! List all active processes
    SHOW PROCESS/PCB/PHD/REG            ! Display current process
   EXIT
$ SET FILE/NOBACKUP SYS$SYSTEM:SAVEDUMP.DMP
```

15.10 Freeing Dump Information from the Page File

If you use SYS$SYSTEM:PAGEFILE.SYS to store a system crash dump, you must later free the space occupied by the dump for use by the pager. If you do not, your system may hang because it has insufficient paging space.

Section 15.1 explains when you might use the page file to store a system crash dump.

How to Perform This Task

1. Invoke the System Dump Analyzer utility (SDA), specifying PAGEFILE.SYS as the target:

   ```
   $ ANALYZE/CRASH_DUMP SYS$SYSTEM:PAGEFILE.SYS
   ```

2. Enter the SDA command COPY in the following format to copy the dump from SYS$SYSTEM:PAGEFILE.SYS to another file:

   COPY dump_filespec

   For example, to copy the dump file off the system disk to a file called SAVEDUMP.DMP on DISK$USER5, enter the following command:

   ```
   SDA> COPY DISK$USER5:[DUMPS]SAVEDUMP.DMP
   ```

   Because a system dump file can contain privileged information, you should protect copies of dump files from world read access.

   To prevent the system from backing up the complete contents of the file, assign the NOBACKUP attribute to the file with the DLC command SET FILE/NOBACKUP.

   Alternatively, to free the pages in the page file that are taken up by the dump without having to copy the dump elsewhere, issue the ANALYZE/CRASH_DUMP/RELEASE command. This command immediately releases the pages to be used for system paging, effectively deleting the dump. Note that this command does *not* allow you to analyze the dump before deleting it.

3. Enter the EXIT command to exit SDA.

4. Include the SDA command entered in steps 1 and 2 in the site-specific startup command procedure SYSTARTUP_VMS.COM to free page space each time the system reboots.

Although the DCL COPY command can also be used to copy a dump file, only the SDA COPY command causes the pages occupied by the dump to be freed from the system's page file.

Example

The following commands, added to SYSTARTUP_VMS.COM command procedure, copy the contents of the page file to a file named SAVEDUMP.DMP:

```
$ ANALYZE/CRASH_DUMP SYS$SYSTEM:PAGEFILE.SYS
  COPY DISK$USER5:[DUMPS]SAVEDUMP.DMP
  EXIT
$ SET FILE/NOBACKUP SYS$SYSTEM:SAVEDUMP.DMP
```

15.11 Creating Page and Swap Files

Primary page and swap files are provided in your distribution kit in the following locations:

SYS$SYSTEM:PAGEFILE.SYS
SYS$SYSTEM:SWAPFILE.SYS

For performance or disk space reasons, you might want to create page and swap files on disks other than the system disk. The following sections explain how to create page and swap files using different methods:

Method	For More Information
Using AUTOGEN (the recommended method)	Section 15.11.1
Using SYSGEN	Section 15.11.2

15.11.1 Using AUTOGEN (Recommended Method)

You can direct AUTOGEN to create new page and swap files by adding symbols to MODPARAMS.DAT to specify the name, location, and size of new files to be created, and running AUTOGEN. Before performing this task, you should understand AUTOGEN and its parameter file MODPARAMS.DAT. For more information, see Section 14.4 and Section 14.4.4.

You can also define symbols in MODPARAMS.DAT to control the size of page, swap, and dump files. For more information, see Section 15.14.1.

How to Perform This Task

1. Add the following symbols to MODPARAMS.DAT to specify the names and locations of the page and swap files to be created:

Definition	For Page Files	For Swap Files
File name and location	PAGEFILE*n*_NAME = *file-spec*	SWAPFILE*n*_NAME = *file-spec*

   For *n*, use an integer that specifies the page or swap file. Refer to the primary page and swap files by specifying a value of 1 for *n*; refer to subsequent files by specifying increasingly higher integer values for *n*. For example, to refer to a secondary page or swap file, specify a value of 2 for *n*.

   For *file-spec*, specify the full file specification of the file to be created.

2. Enter the following command to invoke a first pass of AUTOGEN. In this pass, AUTOGEN displays its calculations for system file sizes to SYS$OUTPUT:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES
```

3. If the file sizes displayed in step 2 are inadequate, add the following symbols to MODPARAMS.DAT to control the size of the files, and return to step 2:

Definition	For Page Files	For Swap Files
File size	MIN_PAGEFILE*n*_SIZE = *block-size*	MIN_SWAPFILE*n*_SIZE = *block-size*

For *n*, specify an integer that indicates the page or swap file. Refer to the primary page and swap files by specifying a value of 1 for *n*; refer to subsequent files by specifying increasingly higher integer values for *n*. For example, to refer to a secondary page or swap file, specify a value of 2 for *n*.

For *block-size*, specify the size in blocks.

4. When you are satisfied with the file sizes displayed in step 2, execute a second pass of AUTOGEN using the following command to install the modified system files when the system is rebooted:

```
$ @SYS$UPDATE:AUTOGEN GENPARAMS REBOOT
```

5. Add commands to the site-specific startup command procedure SYPAGSWPFILES.COM to make sure the files are installed each time the system boots. For instructions, see Section 15.12.

Example

To direct AUTOGEN to create a new secondary swap file named PAGED$:[PAGESWAP]SWAPFILE.SYS that holds 30,000 blocks, add the following symbols to MODPARAMS.DAT:

```
MIN_SWAPFILE2_NAME = "PAGED$:[PAGESWAP]SWAPFILE.SYS"
MIN_SWAPFILE2_SIZE = 30000
```

15.11.2 Using SYSGEN

AUTOGEN is the recommended method for creating page and swap files. However, in an emergency, you can use the System Generation utility (SYSGEN) to directly create files. For example, if you see that page file space is becoming dangerously low, you might use SYSGEN to quickly add page file space to prevent the system from hanging.

How to Perform This Task

1. Determine the names, locations, and sizes of the files you plan to create. For information on determining appropriate sizes, see Section 15.4.

2. Invoke SYSGEN by entering the following command:

```
$ RUN SYS$SYSTEM:SYSGEN
```

3. Enter the SYSGEN command CREATE in the following format:

   CREATE file-spec/SIZE=block-size

   For example:

```
SYSGEN> CREATE DUA2:[PAGE_SWAP]PAGEFILE_1.SYS/SIZE=100000
SYSGEN> CREATE DUA2:[PAGE_SWAP]SWAPFILE_1.SYS/SIZE=100000
```

   If the file you specify as *file-spec* does not exist, this command creates a file by that name that can be used as a page or swap file. If the file does exist, the command does one of the following:

   - If the size you specify is larger than the existing file, the command extends the file.
   - If the size you specify is smaller, the command creates a new, smaller file.

   For more information on the SYSGEN command CREATE, see the SYSGEN section in the *OpenVMS System Management Utilities Reference Manual*.

4. Install the files, following the instructions in Section 15.12. The system automatically installs the primary page and swap files located in SYS$SYSTEM. However, other page files are not automatically installed.
5. Add commands to SYS$MANAGER:SYPAGSWPFILES.COM to install the files each time the system boots. Follow the instructions in Section 15.12.2.
6. If you do not want AUTOGEN to resize the files according to its calculations, edit MODPARAMS.DAT to specify the sizes of these files. Follow the instructions in Section 15.14.1.1.

Example

The following example uses SYSGEN to create page and swap files. It also installs the files as explained in Section 15.12.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE DUA2:[PAGE_SWAP]PAGEFILE_1.SYS/SIZE=100000
SYSGEN> CREATE DUA2:[PAGE_SWAP]SWAPFILE_1.SYS/SIZE=100000
SYSGEN> INSTALL DUA2:[PAGE_SWAP]PAGEFILE_1.SYS/PAGEFILE
SYSGEN> INSTALL DUA2:[PAGE_SWAP]SWAPFILE_1.SYS/SWAPFILE
```

15.12 Installing Page and Swap Files

The system automatically installs the primary page and swap files located in SYS$SYSTEM. However, other page and swap files are not automatically installed. For this reason, if you create secondary page and swap files, you must also install them with the System Generation utility (SYSGEN). Note that SYSGEN INSTALL commands perform a different function than Install utility (INSTALL) commands.

15.12.1 Installing Interactively

1. Invoke SYSGEN by entering the following command:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   ```

2. Enter the SYSGEN command INSTALL in the following format:

   INSTALL file-spec/filetype

   For example:

   ```
   SYSGEN> INSTALL DUA2:[PAGE_SWAP]PAGEFILE_1.SYS/PAGEFILE
   SYSGEN> INSTALL DUA2:[PAGE_SWAP]SWAPFILE_1.SYS/SWAPFILE
   ```

3. To make sure the files are installed each time the system boots, edit SYS$MANAGER:SYPAGSWPFILES.COM to add the commands you entered in step 2. For more information, see Section 15.12.2.

Example

The following example installs page and swap files interactively:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> INSTALL DUA2:[PAGE_SWAP]PAGEFILE_1.SYS/PAGEFILE
SYSGEN> INSTALL DUA2:[PAGE_SWAP]SWAPFILE_1.SYS/SWAPFILE
```

15.12.2 Installing in SYPAGSWPFILES.COM

Page and swap files other than SYS$SYSTEM:PAGEFILE.SYS and SYS$SYSTEM:SWAPFILE.SYS must be reinstalled each time the system boots. You can do this by adding the commands to install the files to the startup command procedure SYS$MANAGER:SYPAGSWPFILES.COM. The template file SYS$MANAGER:SYPAGSWPFILES.TEMPLATE includes comments that help explain how this file is used.

Before performing this task, you must have created the secondary files, as explained in Section 15.11.

For more information on SYPAGSWPFILES.COM, see Section 5.2.3.

You can also use SATELLITE_PAGE.COM to install page and swap files on a VAXcluster or VMScluster satellite node's local disk. SATELLITE_PAGE.COM is created when you run CLUSTER_CONFIG.COM. For more information on installing page and swap files on a satellite node's local disk, see the *VMScluster Systems for OpenVMS* manual.

How to Perform This Task

1. Invoke any editor to edit SYS$MANAGER:SYPAGSWPFILES.COM.
2. If necessary, add a MOUNT command for each disk that holds a page or swap file. This is necessary because only the system disk is mounted at the time SYPAGSWPFILES.COM is invoked.

   For example:

   ```
   $ MOUNT/SYSTEM/NOASSIST DUA2: DISK_SYS2
   ```

   For information on the MOUNT command, see the *OpenVMS DCL Dictionary*.

   The following commands, inserted before the MOUNT command, are also useful to determine if the disk is available before mounting. Note, however, that if the disk is broken and cannot mount, these commands will cause an infinite loop.

   ```
   $ LOOP1:
   $  ON WARNING THEN GOTO LOOP1
   $  WAIT 0000 00:00:00.50
   $  READY = F$GETDVI("device:","AVL")
   $  IF READY .EQS. "FALSE" THEN GOTO LOOP1
   ```

   For *device:*, specify the device name.
3. Add the following command to invoke SYSGEN:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   ```
4. Add commands in the following format to SYPAGSWPFILES.COM to install the files each time the system boots.

   For page files, use the following format:

   INSTALL file-spec/PAGEFILE

   For example:

   ```
   INSTALL DUA2:[SYSTEM]PAGEFILE_1.SYS/PAGEFILE
   ```

   For swap files, use the following format:

   INSTALL file-spec/SWAPFILE

For example:

```
INSTALL DUA2:[SYSTEM]SWAPFILE_1.SYS/SWAPFILE
```

5. Add an EXIT command to exit SYSGEN:

```
EXIT
```

Example

The following example shows commands you might add to SYPAGSWPFILES.COM to install page and swap files named PAGEFILE_1.SYS and SWAPFILE_1.SYS located on the DUA2: device:

```
$ EDIT SYS$MANAGER:SYPAGSWPFILES.COM
  [add the following commands to SYPAGSWPFILES.COM:]
    .
    .
    .
$ MOUNT/SYSTEM/NOASSIST DUA2: DISK_SYS2
$ RUN SYS$SYSTEM:SYSGEN
  INSTALL DUA2:[SYSTEM]PAGEFILE_1.SYS /PAGEFILE
  INSTALL DUA2:[SYSTEM]SWAPFILE_1.SYS /SWAPFILE
  EXIT)
```

15.13 Removing Page, Swap, and Dump Files

Caution

If you need to remove a page, swap, or dump file, do not simply delete the file.

How to Perform This Task

1. Use the RENAME command to rename the file to be deleted.
2. Shut down and reboot the system.
3. Delete the file.
4. When you delete a file, make sure you remove from SYPAGESWPFILES.COM and MODPARAMS.DAT any command lines related to the file.

Example

```
$ RENAME DUA2:[SYSTEM]PAGEFILE_1.SYS; DUA2:[SYSTEM]JUNK.SYS;
$ @SYS$SYSTEM:SHUTDOWN.COM
    .
    .
    .
[SHUTDOWN.COM shuts down and reboots the system]
[When the system reboots, log in]
    .
    .
    .
$ DELETE DUA2:[SYSTEM]JUNK.SYS;
```

15.14 Changing Page, Swap, and Dump File Sizes

The following sections explain how to change sizes of page, swap, and dump files using different methods:

Method	For More Information
Using AUTOGEN (recommended method)	Section 15.14.1
Using SWAPFILES.COM (for primary files only)	Section 15.14.2
Using SYSGEN	Section 15.14.3

15.14.1 Using AUTOGEN (Recommended Method)

AUTOGEN automatically calculates appropriate sizes for page, swap, and dump files. It also modifies the files to the appropriate sizes and installs them. You can control sizes calculated by AUTOGEN by defining symbols in the file MODPARAMS.DAT. For more information, see Section 15.14.1.1.

How to Perform This Task

To change page, swap, and dump files, execute AUTOGEN in two passes as follows:

1. Enter the following command to invoke a first pass of AUTOGEN. In this pass, AUTOGEN displays its calculations for system file sizes to SYS$OUTPUT:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES
```

2. If the file sizes displayed in step 1 are inadequate, add symbols to MODPARAMS.DAT to control the size of files as explained in Section 15.14.1.1 and return to step 1.

3. When you are satisfied with the file sizes displayed in step 1, execute a second pass of AUTOGEN using the following command to install the modified system files when the system is rebooted:

```
$ @SYS$UPDATE:AUTOGEN GENPARAMS REBOOT
```

15.14.1.1 Controlling the Size of Page, Swap, and Dump Files in MODPARAMS.DAT

You can add information to the AUTOGEN parameter file MODPARAMS.DAT to control the sizes that AUTOGEN calculates for page, swap, and dump files. If you do not supply system file size information in MODPARAMS.DAT, AUTOGEN performs default size calculations for page, swap, and dump files.

For information on AUTOGEN, see Section 14.4. For more information on MODPARAMS.DAT, see Section 14.4.4.

You can define symbols in MODPARAMS.DAT to specify either of the following:

Size to Be Specified	For More Information
Total desired size for all page or swap files on a system (not valid for the dump file)	Table 15–5
Sizes for *individual* page, swap, or dump files	Table 15–6

Note

You cannot specify sizes for both total and individual files. AUTOGEN issues a warning if conflicting symbol definitions exist in MODPARAMS.DAT.

For page and swap files, AUTOGEN generally manipulates the primary files SYS$SYSTEM:PAGEFILE.SYS and SYS$SYSTEM:SWAPFILE.SYS *only* if you have no other page and swap files; if you have secondary files, AUTOGEN manipulates the secondary files and excludes primary files. However, in some instances, AUTOGEN might modify the size of the primary page and swap files. If you do not want AUTOGEN to change the sizes of the primary files, specify the following symbols in MODPARAMS.DAT:

```
PAGEFILE = 0
SWAPFILE = 0
```

These symbols direct AUTOGEN to ignore the primary page and swap files when calculating sizes.

If the creation or extension of a file would cause the target disk to become more than 95 percent full, AUTOGEN issues a warning and does not perform the operation.

You can use AUTOGEN to create a page, swap, or dump file that is smaller than the current version of the file. After you have booted and begun using the new file, remember to use the DCL command PURGE to reclaim the disk space from the old version of the file. To determine the current sizes of installed page and swap files, enter the DCL command SHOW MEMORY/FILES. If you have increased the size of any of these files, but you have not yet rebooted, this command displays the original sizes.

Note

AUTOGEN will not change file sizes if you specify a value of 0 or a value that is within 10 percent of the current size.

Table 15–5 lists the symbols you can define in MODPARAMS.DAT to control *total* size of page file, swap file, or dump file space.

Table 15–5 Symbols for Controlling the Total Size of Page, Swap, or Dump File Space

Operation	Page File Symbol	Swap File Symbol	Dump File Symbol
To define the total amount of space	PAGEFILE = *n*[1]	SWAPFILE = *n*[1]	DUMPFILE = *n*[1]
To increase total size	ADD_PAGEFILE = *n*	ADD_SWAPFILE = *n*	ADD_DUMPFILE = *n*
To specify maximum total size	MAX_PAGEFILE = *n*	MAX_SWAPFILE = *n*	MAX_DUMPFILE = *n*

[1] *n* is the total size, in blocks. If *n* is 0, the corresponding AUTOGEN section is skipped. For page and swap files, if *n* is not 0 and no secondary files exist, AUTOGEN applies the value to primary files. If *n* is not 0, and secondary files exist, AUTOGEN applies any change evenly across all secondary page or swap files but, in most cases, does not change primary files.

(continued on next page)

Table 15–5 (Cont.) Symbols for Controlling the Total Size of Page, Swap, or Dump File Space

Operation	Page File Symbol	Swap File Symbol	Dump File Symbol
To specify minimum total size	MIN_PAGEFILE = *n*	MIN_SWAPFILE = *n*	MIN_DUMPFILE = *n*

Table 15–6 lists the symbols you can define in MODPARAMS.DAT to control the size of *individual* files.

Table 15–6 Symbols for Controlling the Size of Individual Page and Swap Files

Operation	Page File Symbol[1]	Swap File Symbol[1]
To specify file size	PAGEFILE*n*_SIZE = *block-size*	SWAPFILE*n*_SIZE = *block-size*
To increase file size	ADD_PAGEFILE*n*_SIZE = *block-size*	ADD_SWAPFILE*n*_SIZE = *block-size*
To specify maximum file size	MAX_PAGEFILE*n*_SIZE = *block-size*	MAX_SWAPFILE*n*_SIZE = *block-size*
To specify minimum file size	MIN_PAGEFILE*n*_SIZE = *block-size*	MIN_SWAPFILE*n*_SIZE = *block-size*

[1]For *n*, specify an integer that indicates the page or swap file. Refer to the primary page and swap files by specifying a value of 1 for *n*; refer to subsequent files by specifying increasingly higher integer values for *n*. For example, to refer to a secondary page or swap file, specify a value of 2 for *n*. For *block-size*, specify the size in blocks.

Examples

The following line in MODPARAMS.DAT specifies that all page file space should total 100000 blocks:

```
PAGEFILE = 100000
```

If you had only a primary page file, the resulting size of that file would be 100000 blocks. If you had multiple page files, the difference between the total current size and the total new size would be spread across secondary files. For example, if you specified PAGEFILE = 100000, the changed page file sizes would be as follows:

File	Original Size (in Blocks)	Resulting Size (in Blocks)
Primary page file	10000	10000
Secondary page file 1	30,000	45000
Secondary page file 2	30,000	45000

To direct AUTOGEN to set the primary page file size to 10000 blocks, you would use the following symbol definition:

```
PAGEFILE1_SIZE = 10000
```

To direct AUTOGEN to create a new secondary swap file named PAGED$:[PAGESWAP]SWAPFILE.SYS that holds 30,000 blocks, use the following symbol definitions:

```
SWAPFILE2_NAME = "PAGED$:[PAGESWAP]SWAPFILE.SYS"
MIN_SWAPFILE2_SIZE = 30000
```

15.14.2 Using SWAPFILES.COM

Digital recommends you use AUTOGEN to change sizes of page, swap, and dump files. However, you can use the command procedure SYS$UPDATE:SWAPFILES.COM to change the size of *primary* page, swap, and dump files. SWAPFILES.COM shows you the current size of the page, swap, and dump files before you change the sizes.

If you change the sizes of page, swap, or dump files, you must be sure to edit MODPARAMS.DAT to specify the new sizes, as explained in Section 15.14.1.1. If you do not specify the new sizes in MODPARAMS.DAT, AUTOGEN will resize the files next time it runs.

The procedure displays the sizes of the current page swap, and dump files in SYS$SYSTEM, and the amount of space remaining on the system disk. It then allows you to enter new sizes, or keep the existing sizes for these files. If you specify a size that is larger than that of an existing file, the procedure automatically extends the size of a page or dump file. If you specify a smaller size for a system page, swap, or dump file, a new version of the file is created.

How to Perform This Task

1. Enter the following command to invoke the command procedure:

   ```
   $ @SYS$UPDATE:SWAPFILES.COM
   ```

   The system displays the current files found in SYS$SYSTEM and their sizes. For example:

   ```
   Current file sizes are:

   Directory SYS$SYSROOT:[SYSEXE]

   PAGEFILE.SYS;1    16384
   SYSDUMP.DMP;1      4128
   SWAPFILE.SYS;1     3072

   Total of 3 files, 23584 blocks.

   There are 128741 available blocks on SYS$SYSDEVICE.
   ```

2. In response to the following prompt, type the desired size, in blocks, for the page file. To keep the same size, press Return:

   ```
   Enter new size for page file:
   ```

3. In response to the following prompt, type the desired size, in blocks, for the dump file. To keep the same size, press Return:

   ```
   Enter new size for system dump file:
   ```

4. In response to the following prompt, type the desired size, in blocks, for the swap file. To keep the same size, press Return:

   ```
   Enter new size for swap file:
   ```

5. Shut down and reboot the system to use the new files.
6. After the system reboots, purge obsolete copies of the files. Do not delete the old files until the system reboots.
7. Edit MODPARAMS.DAT to include the new file sizes, as explained in Section 15.14.1.1. If you do not specify the new sizes in MODPARAMS.DAT, AUTOGEN will automatically resize the files the next time it runs.

Example

```
$ @SYS$UPDATE:SWAPFILES
To leave a file size at its current value type a
carriage return in response to its size prompt.
Current file sizes are:

Directory SYS$SYSROOT:[SYSEXE]

PAGEFILE.SYS;1    100000
SYSDUMP.DMP;1      28000
SWAPFILE.SYS;1     33000

Total of 3 files, 161000 blocks.

There are 128741 available blocks on SYS$SYSDEVICE.

Enter new size for page file: [Return]
Enter new size for system dump file: 30000
Enter new size for swap file: [Return]

%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]SYSDUMP.DMP;1 extended

**********************************************************************
*  Please reboot in order for the new files to be used by the system. *
*  After rebooting, purge obsolete copies of the files.              *
*  DO NOT delete the old files until after the reboot.               *
**********************************************************************
```

15.14.3 Using SYSGEN

Digital recommends you use AUTOGEN to create and change page, swap, and dump files. AUTOGEN invokes the System Generation utility (SYSGEN) to create or change the files. However, in an emergency, you can use the System Generation utility (SYSGEN) to directly change the size of page, swap and dump files. For example, if you see that page file space is becoming dangerously low, you might use SYSGEN to quickly add page file space to prevent the system from hanging.

How to Perform This Task

1. Determine the appropriate size of the files. For information, see Section 15.4.
2. Invoke SYSGEN and enter the CREATE command in the following format:

   CREATE file-spec/SIZE=block-size

   For *file-spec*, specify the full file specification.

   For *block-size*, specify the size of the file in blocks.

   If the file you specify already exists and the size you specify is larger than the existing file, the command extends the existing file. If the file you specify already exists and the size you specify is smaller than the existing file, the command creates a new file of the specified size.

   For example, the following command extends the existing, smaller primary page file PAGEFILE.SYS:

   ```
   SYSGEN> CREATE PAGEFILE.SYS/SIZE=100000
   ```

   For more information on the SYSGEN command CREATE, see the SYSGEN section in the *OpenVMS System Management Utilities Reference Manual*.

   Note

   Frequent file creation and deletion can cause the free space on a disk to become severely fragmented. SYSGEN issues a HEADERFULL warning message if it determines that the creation or extension of a system file would cause that file to become fragmented enough to render the system unbootable. If this occurs, Digital recommends that you back up and restore your system disk to consolidate the free space on the volume into one contiguous area. (For more information, see Section 10.17.) After you have restored the disk, retry the SYSGEN operation. In cases where SYSGEN issues a warning message, the file might be somewhat larger, but not as large as the value specified in the CREATE command.

3. Use the following table to determine if you need to reboot to use the new or modified file:

Type	Change	Reboot Required?
Primary page, swap, or dump file[1]	New file	Yes
	Extended file	Yes
Secondary page or swap file	New file	No
	Extended file	Yes

[1]Primary page, swap, and dump files are SYS$SPECIFIC:[SYSEXE] PAGEFILE.SYS, SWAPFILE.SYS, SYSDUMP.DMP.

4. If you create a new version of the file, purge the old version *after* the system reboots.

Example

The commands in the following example extend the existing files PAGEFILE.SYS, SWAPFILE.SYS, and SYSDUMP.DMP to the specified sizes:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE PAGEFILE.SYS/SIZE=100000
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]PAGEFILE.SYS;1 extended
SYSGEN> CREATE SWAPFILE.SYS/SIZE=30000
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]SWAPFILE.SYS;1 extended
SYSGEN> CREATE SYSDUMP.DMP/SIZE=33000
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]SYSDUMP.DMP;1 extended
SYSGEN> EXIT
```

16
Performance Considerations

This chapter introduces the basic concepts of performance management. For more detailed information, see one of the following manuals:

- On VAX systems, see the *Guide to OpenVMS Performance Management.*
- On AXP systems, see *A Comparison of System Management on OpenVMS AXP and OpenVMS VAX.*

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Knowing your work load	Section 16.2
Choosing a workload management strategy	Section 16.3
Distributing work load	Section 16.4
Predicting when tuning is required	Section 16.6
Evaluating tuning success	Section 16.7
Choosing performance options	Section 16.8
Installing images with the Install utility (INSTALL)	Section 16.9

This chapter explains the following concepts:

Concept	Section
Performance management	Section 16.1
System tuning	Section 16.5
Images and known images	Section 16.9.1
Known file lists	Section 16.9.2
Attributes of known images	Section 16.9.3

16.1 Understanding Performance Management

Performance management means optimizing your hardware and software resources for the current work load. This task entails several distinct but related activities:

- Acquiring a thorough familiarity with your work load and an understanding of how that work load exercises the system's resources. This knowledge, combined with an appreciation of the operating system's resource management mechanisms, will enable you to establish realistic standards for system performance in areas such as the following:

  - Interactive and batch throughput
  - Interactive response time
  - Batch job turnaround time
- Routinely monitoring system behavior to determine if, when, and why a given resource is approaching capacity.
- Investigating reports of degraded performance from users.
- Planning for changes in the system work load or hardware configuration and being prepared to make any necessary adjustments to system values.
- Performing, after installation, certain optional system management operations.

16.2 Knowing Your Work Load

One of the most important assets that a system manager brings to any performance evaluation is an understanding of the normal work load and behavior of the system. Each system manager must assume the responsibility for understanding the system's work load sufficiently to be able to recognize normal and abnormal behavior; to predict the effects of changes in applications, operations, or usage; and to recognize typical throughput rates. The system manager should be able to answer such questions as the following:

- What is the typical number of users on the system at any given time of day?
- What is the typical response time for various tasks for this number of users, at any given hour of operation?
- What are the peak hours of operation?
- Which jobs typically run at which time of day?
- Which commonly run jobs are intensive consumers of the CPU, memory, and disk space?
- Which applications involve the most image activations?
- Which parts of the system software, if any, have been modified or user-written, such as device drivers?
- Are there any known system bottlenecks? Are there any anticipated ones?

If you are new to the OpenVMS operating system or to system management, you should observe system operation using the following tools:

- Monitor utility
- Accounting utility
- SHOW commands (available through DCL)

The *Guide to OpenVMS Performance Management* provides detailed procedures for using the Monitor utility and, to a lesser extent, other operating system tools to observe and evaluate system performance.

Over time you will learn about metrics such as the typical page fault rate for your system, the typical CPU usage, the normal memory usage, and typical modes of operation. You will begin to see how certain activities affect system performance and how the number of users or the time of day affects some of the values.

As you continue to monitor your system, you will come to know what range of values is acceptable, and you will be better prepared to use these same tools, together with your knowledge, to detect unusual conditions. Routine evaluation of the system is critical for effective performance management. The best way to avoid problems is to anticipate them; you should not wait for problems to develop before you learn how the system performs.

You can learn more about your system's operation if you use the Monitor and Accounting utilities on a regular basis to capture and analyze certain key data items. By observing and collecting this data, you will also be able to see usage trends and predict when your system may reach its capacity.

You should also understand that system resources are used by system management tools. Be careful, therefore, in selecting the items you want to measure and the frequency with which you collect the data. If you use the tools excessively, the consumption of system resources to collect, store, and analyze the data can distort your picture of the system's work load and capacity. The best approach is to have a plan for collecting and analyzing the data.

16.3 Choosing a Workload Management Strategy

System performance is directly proportional to the efficiency of workload management. Each installation must develop its own strategy for workload management. Before adjusting any system values, make sure you have resolved the following issues and that your workload management strategy is correct:

- [ ] Is there a time of day when the work load "peaks," that is, when it is noticeably heavier than at other times?
- [ ] Is there any way to balance the work load better? Perhaps some voluntary measures can be adopted by users, after appropriate discussion.
- [ ] Could any jobs be run better as batch jobs, preferably during nonpeak hours?
- [ ] Have primary and secondary hours of operation been employed with users? If not, could system performance benefit by adopting this practice? If the primary and secondary hours are in use, are the choices of hours the most appropriate for all users? (Plan to review this issue every time you either add or remove users or applications, to ensure that the desired balance is maintained.)
- [ ] Can future applications be designed to work around any known or expected system bottlenecks? Can present applications be redesigned somewhat, for the same purpose? (See the *Guide to OpenVMS File Applications*.)
- [ ] Are you using to the utmost the code-sharing ability that the operating system offers? If not, you will find that code sharing provides an excellent means to conserve memory, thereby improving performance over the life of the system.

16.4 Distributing the Work Load

You should distribute the work load as evenly as possible over the time your system is running. Although the work schedule for your site may make it difficult to schedule interactive users at optimum times, the following techniques may be helpful:

- Run large jobs as batch jobs—Establish a site policy that encourages the submission of large jobs on a batch basis. Regulate the number of batch streams so that batch usage is high when interactive usage is low. You might also want to use DCL command qualifiers to run batch jobs at lower priority, adjust the working set sizes, or control the number of concurrent jobs. For information about setting up your batch environment, see Section 13.5.

- Restrict system use—Do not permit more users to log in at one time than the system can support with an adequate response time. You can restrict the number of interactive users with the DCL command SET LOGINS /INTERACTIVE. You can also control the number of concurrent processes with the MAXPROCESSCNT system parameter, and the number of remote terminals allowed to access the system at one time with the RJOBLIM system parameter. See Section 14.5 for information about modifying system parameters. See the *OpenVMS System Management Utilities Reference Manual* for descriptions of all system parameters.

  You might also restrict use of the system by groups of users to certain days and hours of the day. You can use the Authorize utility to define the permitted login hours for each user. In particular, refer to the AUTHORIZE qualifiers /PRIMEDAYS, /P_RESTRICT, /PFLAGS, /SFLAGS, and /S_RESTRICT. For more information, see Chapter 6 and the AUTHORIZE section of the *OpenVMS System Management Utilities Reference Manual.*

  You can use the DCL command SET DAY to override the conventional day of the week associations for primary and secondary days. For example, you might need to specify a primary day of the week as a secondary day when it is a holiday.

- Design applications to reduce demand on binding resources—If you know where your system bottlenecks are or where they will likely occur in the near future, you can distribute the work load more evenly by planning usage that minimizes demand on any bottleneck points. (See the *Guide to OpenVMS File Applications.*)

16.5 Understanding System Tuning

Tuning is the process of altering various system values to obtain the optimum *overall* performance possible from any given configuration and work load. However, the process does not include the acquisition and installation of additional memory or devices, although in many cases such additions (when made at the appropriate time) can vastly improve system operation and performance.

Always aim for best overall performance, that is, performance viewed over time. The work load is constantly changing on most systems. System parameters that produce optimal performance at one time may not produce optimal performance a short time later as the work load changes. Your goal is to establish values that, on average, produce the best overall performance.

Before you undertake any action, you must recognize that the following sources of performance problems cannot be cured by adjusting system values:

- Improper operation
- Unreasonable performance expectations
- Insufficient memory for the applications attempted
- Inadequate hardware configuration for the work load, such as too slow a processor, too few buses for the devices, too few disks, and so forth

- Improper device choices for the work load, such as using disks with insufficient speed or capacity
- Hardware malfunctions
- Human errors, such as poor application design or allowing one process to consume all available resources

When you make adjustments, you normally select a very small number of values for change, based on a careful analysis of the behavior being observed. You control system resources by tuning the values of two types of parameters:

Parameter Type	Description
System parameters	The values set for system parameters control system resources on a systemwide basis. The AUTOGEN command procedure automatically sets system parameters to appropriate values for your system configuration. AUTOGEN can also record feedback from a running system to adjust those parameters based on the system's work load. The *Guide to OpenVMS Performance Management* describes how to select the parameters and new values that are likely to produce the desired changes. Section 14.5 explains how to use AUTOGEN to modify system parameter values.
UAF limits and quotas	The values set for limits and quotas in each User Authorization File (UAF) record control system resources on a per-user basis. To control these values, use the Authorize utility. For information, see Section 6.15.

Before you undertake any tuning operation, be sure you are familiar with the resource management mechanisms described in the *Guide to OpenVMS Performance Management* and *A Comparison of System Management on OpenVMS AXP and OpenVMS VAX*. Understand the nature of system values before adjusting them. Without the proper level of understanding, you might very well degrade, rather than improve, overall performance.

Finally, while investigating the cause of an apparent performance problem, keep in mind that tuning is a last resort.

16.6 Predicting When Tuning Is Required

Under most conditions, tuning is rarely required for OpenVMS systems. The AUTOGEN command procedure, which is included in the operating system, establishes initial values for all the configuration-dependent system parameters so that they match your particular configuration. For information about AUTOGEN, see Section 14.4.

Additionally, the system includes features that, in a limited way, permit it to adjust itself dynamically during operation. That is, the system detects the need for adjustment in certain areas, such as the nonpaged dynamic pool, the working set size, and the number of pages on the free and modified page lists. The system makes rough adjustments in these areas automatically. As a result, these areas can grow dynamically, as appropriate, during normal operation.

Experience has shown that the most common cause of disappointment in system performance is insufficient hardware capacity. Once the demand on a system exceeds its capacity, adjusting system values will not result in any significant improvements, simply because such adjustments are a means of trading off or juggling existing resources.

Although tuning is rarely required, you should recognize that system tuning may be needed under the following conditions:

- If you have adjusted your system for optimal performance with current resources and then acquire new capacity, you must plan to compensate for the new configuration. In this situation, the first and most important action is to execute the AUTOGEN command procedure.

  For more information about AUTOGEN, see Section 14.4.
- If you anticipate a dramatic change in your work load, you should expect to compensate for the new work load.

16.7 Evaluating Tuning Success

Whenever you adjust your system, you should monitor its behavior afterward to be sure that you have obtained the desired results. To observe results, use the Monitor utility and the various forms of the DCL command SHOW. See the *OpenVMS DCL Dictionary* for detailed information on the SHOW command. See Section 17.7.2 for information about using MONITOR. See the *OpenVMS System Management Utilities Reference Manual* (MONITOR) for detailed descriptions of MONITOR commands.

For example, you might consider running some programs whose results you believe are fixed and reproducible at the same time that you run your normal work load. If you run the programs and measure their running times under nearly identical workload conditions both before and after your adjustments, you can obtain a basis for comparison.

However, when applying this technique, remember to take the measurements under very similar workload conditions. Also, remember that this test alone does not provide conclusive proof of success. There is always the possibility that your adjustments may have favored the performance of the image you are measuring—to the detriment of other images. Therefore, in all cases, continue to observe system behavior closely for a time after you make any changes.

16.8 Choosing Performance Options

Following is a list of optional system management operations, normally performed after installation, that often result in improved overall performance. Choose the options that are appropriate for your site. Not all options are appropriate at every site.

- [ ] Decompress system libraries—Most of the libraries shipped with the operating system are in a compressed format in order to conserve disk space. The system dynamically decompresses them whenever they are accessed, and the resulting performance slowdown is especially noticeable during link operations and when requesting online help. If you have sufficient disk space, decompressing the libraries improves both CPU and elapsed time performance. To do this, invoke the command procedure SYS$UPDATE:LIBDECOMP.COM. The decompressed object libraries use about 25 percent more disk space than when compressed; the decompressed help libraries use about 50 percent more disk space.

☐ Disable file system high-water marking—This security feature is set by default when a volume is initialized to guarantee that users cannot read data they have not written.

For non-shared sequential files, the performance impact of high-water marking is minimal. However, for files of non-sequential format, high-water marking creates some overhead; the system erases the previous contents of the disk blocks allocated every time a file is created or extended.

Disabling the feature improves system performance by a variable amount, depending on the following factors:

- How frequently new files are created
- For indexed and relative files, how frequently existing files are extended
- How fragmented the volume is

Be sure to consider the security implications before you disable high-water marking.

To disable high-water marking, you can specify the /NOHIGHWATER qualifier when initializing the volume, or you can disable high-water marking with the DCL command SET VOLUME in the following format:

```
SET VOLUME/NOHIGHWATER_MARKING device-spec[:]
```

☐ Set file extend parameters for OpenVMS Record Management Services (RMS)—Because files extend in increments of twice the multiblock count (default 16), system defaults provide file extension of 32 blocks rounded up to the nearest multiple of the disk's cluster size. Thus, when files are created or extended, increased I/O may slow performance. The problem can be corrected by specifying larger values for file extend parameters or by setting the system parameter RMS_EXTEND_SIZE. See Section 14.5 for information about modifying system parameters. See the *OpenVMS System Management Utilities Reference Manual* for a description of all system parameters.

For more information about establishing the file extension quantity, see the section on tuning in the *Guide to Creating OpenVMS Modular Procedures*.

VAX

☐ On VAX systems, relink images— Beginning with VAX/VMS Version 4.0, the Run-Time Library (VMSRTL) was separated into five smaller libraries. Running images linked under previous versions of the operating system will therefore incur the image activation costs of mapping all five libraries, even if only one is needed. You may improve performance by relinking pre-Version 4.0 images that reference run-time library routines, so that only the required libraries are mapped and activated. ♦

☐ Install frequently used images—When an image is accessed concurrently by more than one process on a routine basis, install the image with the Install utility (INSTALL), specifying the /OPEN, /SHARED, and /HEADER_RESIDENT qualifiers. You will thereby ensure that all processes use the same physical copy of the image, and that the image will be activated in the most efficient way.

Generally, an image takes about two additional physical pages when installed with the /OPEN, /HEADER_RESIDENT, and /SHARED qualifiers. The utility's LIST/FULL command shows the highest number of concurrent accesses to an image installed with the /SHARED qualifier. This information can help you decide whether installing an image is an efficient use of memory.

See Section 16.9.11 and the INSTALL section of *OpenVMS System Management Utilities Reference Manual* for more information on installing images.

AXP

- ☐ On AXP systems, install shareable and executable images specifying the /RESIDENT qualifier with the Install utility. For more information, see Section 16.9.6. ♦

- ☐ Reduce system disk I/O—You can move frequently accessed files off the system disk and use logical names to specify the location or, where necessary, other pointers to access them. For example:
  - SYSUAF.DAT (SYSUAF is the logical name)
  - RIGHTSLIST.DAT (RIGHTSLIST is the logical name)
  - VMSMAIL.DAT (VMSMAIL is the logical name)
  - NETPROXY.DAT (NETPROXY is the logical name)
  - The queue database (for more information, see Section 12.3)
  - ERRFMT log files (SYS$ERRORLOG is the logical name)
  - MONITOR log files (SYS$MONITOR is the logical name)
  - The accounting log file (ACCOUNTNG is the logical name)
  - SECURITY_AUDIT.AUDIT$JOURNAL (SET AUDIT /JOURNAL=SECURITY/DESTINATION= *filespec*)
  - Default DECnet for OpenVMS accounts (records included in the SYSUAF file on the OpenVMS distribution kit)

  To redefine logical names for these system files, edit the site-specific command procedure SYS$MANAGER:SYLOGICALS.COM. For more information on defining logical names in SYLOGICALS.COM, see Section 5.2.5.

  You can also consider moving paging and swapping activity off the system disk by creating large secondary page and swap files on a less heavily used disk. However, if you want to store crash dumps for diagnosing system failures, the dump file must reside in the system-specific directory SYS$SPECIFIC:[SYSEXE] on the system disk for storing crash dumps; if no dump file exists in SYS$SPECIFIC:[SYSEXE], the primary page file must be located there if you want to store crash dumps. For detailed information on moving page and swap files, see Section 15.11.

16.9 Using INSTALL to Install Known Images

The Install utility (INSTALL) stores information about images in memory. Use INSTALL for the following reasons:

Reason	For More Information
To conserve memory use for images that are used concurrently	Section 16.9.4
To improve system performance	Section 16.9.5

Reason	For More Information
‡On AXP systems, with sliced images to improve performance	Section 16.9.6
To make programs that require enhanced privileges available for general use	Section 16.9.7
To allow a nonprivileged process to perform the privileged functions of the image	Section 16.9.7
To mark a sharable image as trusted so it can be invoked by privileged executable images	Section 16.9.7

‡AXP specific

The site-independent startup command procedure, STARTUP.COM, uses INSTALL to install certain system images when the system boots. You use INSTALL to install other selected images, according to the needs of your site.

Installed images must be reinstalled each time the system reboots. To do so, include INSTALL commands in the site-specific startup command procedure SYSTARTUP_VMS.COM, as explained in Section 5.2.7.

Note that Install utility (INSTALL) commands perform a different function than System Generation utility (SYSGEN) INSTALL commands.

The following sections explain installed images and how to use the Install utility.

16.9.1 Understanding Images and Known Images

An **image** is a collection of procedures and data bound together by the Linker utility to form an executable program. Executable programs can be executed (or run) by a process. Usually, executable programs have the file type .EXE.

There are three types of images:

Image Type	Description
Executable	An image linked with the /EXECUTABLE qualifier (or without the /SHAREABLE qualifier) of the Linker utility. For more information, see the *OpenVMS Linker Utility Manual.*
Shareable	An image linked with the /SHAREABLE qualifier of the Linker utility; it must subsequently be linked into an executable image to be used. (Shareable images are sometimes referred to as linkable images, because they can be specified—implicitly or explicitly—as input files to the link of another file.) A shareable image is not copied into the executable images that link with it. Thus, only one copy of the shareable image need be on disk, no matter how many executable images have linked with it. For more information, see the *OpenVMS Linker Utility Manual.*
System	An image that does not run under the control of the operating system. It is intended for standalone operation only. The content and format of a system image differs from that of shareable images and executable images. For more information, see the *OpenVMS Linker Utility Manual.*

When you install an image with INSTALL, the image is assigned attributes and becomes known to the system. For this reason, an installed image is also called a **known image**.

The DCL command RUN parses search lists in a manner that favors known images. On its first pass through the search list, the RUN command looks up images on known file lists and executes each known image that it finds. On its second pass through the search list, the RUN command looks up images on disk and executes those images not executed in the first pass.

16.9.2 Understanding Known File Lists

The system defines known images in internal data structures called **known file lists**. Each entry in the known file list identifies the file name of the installed file and the attributes with which it was installed (for information about attributes of installed images, see Section 16.9.3).

A separate known file list exists for all installed images whose device, directory, and file type are identical. For example, all installed images with the file name DISK$VOLUME:[MAIN]*filename*.EXE would be in one known file list, and all installed images with the file name DISK$VOLUME:[TEST]*filename*.EXE would be in another known file list.

Known file lists last only while the system is operating. If the system is shut down or fails for any reason, you must reinstall all known images after the system is rebooted.

16.9.3 Understanding the Attributes You Can Assign to Known Images

By specifying appropriate qualifiers to INSTALL commands, you can assign attributes to known images. Table 16–1 describes these attributes and the qualifiers that are used to assign them to known images.

Table 16–1 Attributes of Known Images

Attribute	Description	Qualifier
Header resident	The header of the image file (native images only) remains permanently resident, saving one disk I/O operation per file access. For images with single-block file headers, the cost is less than 512 bytes of paged dynamic memory per file; for images with multiblock headers, the cost varies according to the header block count. The images must also be declared permanently open.	/[NO]HEADER_RESIDENT
Permanently open	Directory information on the image file remains permanently resident, eliminating the usual directory search required to locate a file. The cost of keeping an image file permanently open is approximately 512 bytes of paged dynamic memory per file.	/OPEN
Privileged	Amplified privileges are temporarily assigned to any process running the image, permitting the process to exceed its user authorization file (UAF) privilege restrictions during execution of the image. In this way, users with normal privileges can run programs that require higher-than-normal privileges. This attribute (and the /PRIVILEGED qualifier that creates it) applies only to executable images.	/PRIVILEGED[=(privilege,...)]
Protected	A shareable image contains protected code, that is, code that runs in kernel or executive mode but that can be called by a user-level image. Protected images must be declared shareable.	/PROTECTED

(continued on next page)

Table 16–1 (Cont.) Attributes of Known Images

Attribute	Description	Qualifier
‡Resident	On AXP systems, improves the performance of shareable or executable images that have been linked with /SHARE and a new LINK qualifier, /SECTION_BINDING=CODE, by installing them as resident with the Install utility. The code sections of an installed resident shareable image reside in huge pages called granularity hint regions (GHRs) in memory. The AXP hardware can consider a set of pages as a single GHR. This GHR can be mapped by a single page table entry (PTE) in the translation buffer (TB). The result is a reduction in TB miss rates. For more information, see Section 16.9.6.	/RESIDENT
Shareable	More than one user can access the read-only and non-copy-on-reference read/write sections of the image concurrently, so that only one copy of those sections ever needs to be in physical memory. (Copy-on-reference sections always require a separate copy for each process.) The image is implicitly declared permanently open.	/SHARED
Writable	When a shareable non-copy-on-reference writable section is removed from physical memory (for paging reasons or because no processes are referencing it), it is written back to the image file. Any updates made by processes mapped to the section, therefore, are preserved (while the initial values are lost). The image must also be declared shareable.	/WRITABLE

‡AXP specific

16.9.4 Installing Images to Conserve Memory

Shareable images conserve memory because only one copy of the code needs to be in memory at any time, and many users can access the code concurrently. The Install utility is the only way to install images. Use the /SHARED qualifier to install images as shareable images.

When you install a shareable image, more than one user can access the read-only and non-copy-on-reference read/write sections of the image concurrently, so that only one copy of those sections ever need be in physical memory. (Copy-on-reference sections always require a separate copy for each process.) The image is implicitly declared permanently open.

When you install an image with the shareable attribute, permanent system global sections are created. Execution of non-copy-on-reference global sections requires only one copy per section to be in physical memory, no matter how many processes are running the image to which the sections belong.

The number of images you can install with the shareable attribute is restricted by the GBLPAGES and GBLSECTIONS system parameters. For more information on these system parameters, see the *OpenVMS System Management Utilities Reference Manual.*

When an image is not installed, or is installed without the shareable attribute, each process running the image requires private sections in memory.

A shareable image linked to an executable image need not be installed to be executed. At image execution time, the system creates private sections from the shareable image. The only exception is that a shareable image containing a writable non-copy-on-reference section must be installed as a known image with the shareable and writable attributes.

16.9.5 Installing Images to Improve Image Performance

Image performance improves when programs are installed, because the operating system opens any installed file by file ID rather than by file name, thus eliminating costly directory operations.

Installing images as header resident further enhances performance because the system avoids the overhead of I/O operations to read the image header into memory.

VAX

Note

On VAX systems, Virtual I/O Cache can automatically improve image performance at a level similar to that gained by installing images. With Virtual I/O Cache, you might not need to also install images. However, you should decide whether to install based on the configuration and requirements of your site. For more information on Virtual I/O Cache, see the *Guide to OpenVMS Performance Management*. ♦

To install an image as header resident, specify the /HEADER_RESIDENT qualifier when you install the image. This makes the header of the image file (native images only) remain permanently resident, saving one disk I/O operation per file access. For images with single-block file headers, the cost is less than 512 bytes of paged dynamic memory per file; for images with multiblock headers, the cost varies according to the header block count. The images must also be declared permanently open by specifying the /OPEN qualifier.

Frequently accessed images, critical to a site's operations, can be installed as open images. To install an image as permanently open, specify the /OPEN qualifier when you install the image. This makes the directory information on the image file remain permanently resident, eliminating the usual directory search required to locate a file. The cost of keeping an image file permanently open is approximately 512 bytes of paged dynamic memory per file.

16.9.6 Installing Resident Images to Improve Performance (AXP Only)

AXP

On AXP systems, you can improve the performance of shareable images that have been linked with /SHARE and a new LINK qualifier, /SECTION_BINDING=CODE, by installing them as resident with the Install utility. The code sections of an installed resident shareable image reside in huge pages called granularity hint regions (GHRs) in memory. The AXP hardware can consider a set of pages as a single GHR. This GHR can be mapped by a single page table entry (PTE) in the translation buffer (TB). The result is a reduction in TB miss rates.

This feature lets the operating system split the contents of images and sort the pieces so that they can be placed with other pieces that have the same page protection in the same area of memory. Consequently, TBs on AXP systems are used more efficiently than if the images were loaded in the traditional manner.

Application programmers are the likely users of the slicing feature for shareable images. As system manager, you might be asked to coordinate or assist slicing efforts by installing images as resident shareable images. Specify the /RESIDENT qualifier with the INSTALL commands ADD, CREATE, and REPLACE to install shareable and executable images as resident.

Note

The /RESIDENT qualifier is applicable only to loading shareable images or executable images. ♦

16.9.7 Installing Images to Enhance Privileges of Images

There are two ways to allow an image to execute in an enhanced privilege environment:

- Installing existing executable images with extra privileges to allow a nonprivileged process to perform the privileged functions of the image.

  Use the /PRIVILEGED qualifier for the INSTALL commands ADD or CREATE.
- Installing privileged shareable images (which are used to implement user-written system services), allowing other, nonprivileged images to execute select portions of privileged code without enhancing the privileges of those individual images.

  Use the /PROTECTED and /SHARED qualifiers for the INSTALL commands ADD or CREATE.

Caution

Installing an image with enhanced privilege can compromise system security. Make sure the image does not enable a user to regain control with extra privileges enabled.

16.9.7.1 Privileged Executable Images

A nonprivileged process can perform the privileged functions of an executable image when it is installed as a privileged image. Install executable images with enhanced privileges by using the /PRIVILEGED qualifier; amplified privileges are temporarily assigned to any process running the image (executable images only), permitting the process to exceed its user authorization file (UAF) privilege restrictions during execution of the image. In this way, users with normal privileges can run programs that require higher-than-normal privileges.

For an image installed with privileges to activate another image, such as a shareable image, either by having it linked to the privileged image or by using LIB$FIND_IMAGE_SYMBOL, the following conditions hold:

- The shareable image must be installed as a known image using INSTALL.

Note

Installing the Install utility itself requires that a number of shareable images have been previously installed. If any of those required shareable images (such as SMG$SHR, LIBOTS, and so on) is unavailable, the execution of the Install utility fails. Since INSTALL will not work in this situation, you cannot simply install the missing images. To work around this problem, redefine the INSTALL command as follows:

```
$ DEFINE INSTALL SYS$SYSTEM:INSTALL.EXE;0
```

When you now enter the INSTALL command, the image activator does not check the known files list for INSTALL.EXE, and the INSTALL command will complete, allowing you to install the required shareable images.

- Logical names and table names used to find the image must be defined in executive or kernel mode. In particular, the standard executive-mode definition of LNM$FILE_DEV translates only to LNM$SYSTEM; definitions in the process, job, or group tables are not recognized.
- Only images linked with the Linker utility qualifiers /NODEBUG and /NOTRACE can be installed with enhanced privilege.

16.9.7.2 Privileged Shareable Images

A user-written system service assumes the privileges it requires when you install it as a privileged shareable image. To create a privileged, shareable image, you must:

1. Link a shareable image with the /PROTECT command qualifier or the PROTECT= option of the Linker utility, so that the image acquires its particular form of enhanced privileges.
   - Use the /PROTECT command qualifier when all parts of an image require protection.
   - Use the PROTECT= option when only part of a privileged shareable image requires protection.
2. Install the privileged shareable image with the Install utility, specifying both the /PROTECTED and the /SHARED qualifiers. The /PROTECTED qualifier assigns the protected attribute. The /SHARED qualifier assigns the shareable attribute. See Section 16.9.3 for information about these attributes.

Note

You cannot create a privileged shareable image using the /PRIVILEGED qualifier for the INSTALL commands ADD or CREATE. This qualifier works only for executable images.

For more information on creating privileged shareable images, see the *OpenVMS Programming Concepts Manual.*

16.9.8 Installing Images to Allow Execution of Images Without Read Access

The image activator allows execution of images to which the caller has execute but not read access.

16.9.8.1 Execute-Only Executable Images

When a process runs an executable image to which it has execute but not read access, the image activator enters a restricted mode of operation similar to that entered when a privileged program is run. In this mode of operation:

- All shareable images activated during the life of the execute-only image must be installed.

- The image activator directs OpenVMS RMS to use only **trusted logical names** (logical names associated with executive or kernel mode) when opening image files.

16.9.8.2 Execute-Only Shareable Images

When an installed execute-only shareable image is run, the image activator enters the same restricted mode of operation that it enters when it detects an execute-only executable image, that is:

- Only trusted logical names are used by OpenVMS RMS.
- All shareable images that the shareable image references must be installed.

In addition, the executable image must be installed with the /EXECUTE_ONLY qualifier, which enables that image to activate shareable images to which the process has execute but not read access.

Note

The /EXECUTE_ONLY qualifier has meaning only for executable images.

16.9.9 Determining Which Images to Install

You should install images that meet the following conditions:

- Images that run frequently
- Images that usually run concurrently from several processes
- Images that require special privileges
- AXP On AXP systems, images that have been linked with the Linker utility qualifiers /SHARE and /SECTION_BINDING=CODE.

  You can use ANALYZE/IMAGE on an AXP system to determine whether an image is linked with /SECTION_BINDING=CODE. In the ANALYZE/IMAGE output, look for the EIHD$V_BIND_CODE symbol; a value of 1 indicates the /SECTION_BINDING=CODE was used. For more information, see the *OpenVMS Linker Utility Manual*. ♦

Because an installed file requires system resources, such as paged dynamic memory, install those files that most improve system performance and site requirements. The INSTALL command LIST provides information about installed images to help you evaluate the merits of installing images. For example, the LIST command calculates the number of times each image is accessed, and shows the number of concurrent accesses, so you can determine if the installation of the images is worth the overhead.

16.9.10 Specifying File Names in INSTALL

When you use INSTALL commands, your file specifications must name existing executable or shareable images. OpenVMS Record Management Services (RMS) resolves each file specification using the following defaults:

- A device and directory type of SYS$SYSTEM
- A file type of .EXE

Unless a file shares these defaults, you must specify a device and directory name and a file type with each file name. The highest existing version of the file is used by default. However, you can specify another version of the file as the known version of the image. Even if other versions of the file exist, the version that you specify will be the version that satisfies all known file lookups for the image.

16.9.11 Installing Images with INSTALL

Before performing this task, you should understand the following:

- Attributes of installed images. For information, see Section 16.9.3.
- File specifications for the Install utility. For information, see Section 16.9.10.

How to Perform This Task

1. Give yourself the CMKRNL privilege by entering the following command:

   ```
   $ SET PROCESS/PRIVILEGES=CMKRNL
   ```

2. Invoke INSTALL by entering the following command:

   ```
   $ INSTALL
   ```

3. Enter the ADD command in the following format:

   ADD file-spec [/qualifier...]

   Specify one or more of the following qualifiers, depending on which attributes you want to assign to the image:

   /EXECUTE_ONLY
   /HEADER_RESIDENT
   /OPEN
   /PRIVILEGED
   /PROTECTED
   /RESIDENT (AXP systems only)
   /SHARED
   /WRITABLE

   For more information on installing images, see the INSTALL command ADD in the INSTALL section of the *OpenVMS System Management Utilities Reference Manual*.

16.9.12 Displaying Known Images with INSTALL

Use the INSTALL command LIST to display information about known images.

The information displayed with the /FULL qualifier of the LIST command can help you determine if installing an image is worth the expense.

How to Perform This Task

1. Invoke INSTALL by entering the following command:

   ```
   $ INSTALL
   ```

2. To display a list of all known images and their attributes, enter the LIST command as follows:

   ```
   INSTALL> LIST
   ```

   To display attributes for a specific image, specify the name of the image as follows:

   LIST filename

For example:

```
INSTALL> LIST LOGINOUT
```

To display complete information about a specific image, including the number of accesses, the number of concurrent accesses, and the number of global sections created, specify the /FULL qualifier as follows:

LIST/FULL filename

Example

The following example displays complete information about the installed image LOGINOUT.EXE, including the number of accesses, the number of concurrent accesses, and the number of global sections created:

```
$ INSTALL
INSTALL> LIST/FULL LOGINOUT

DISK$VMS551:<SYS2.SYSCOMMON.SYSEXE>.EXE
   LOGINOUT;2       Open Hdr Shar Prv
        Entry access count         = 36366
        Current / Maximum shared   = 1 / 10
        Global section count       = 3
        Privileges = CMKRNL SYSNAM LOG_IO ALTPRI TMPMBX SYSPRV
INSTALL>
```

16.9.13 Defining Logical Names for Shareable Image Files

If a shareable image is not located in SYS$SHARE, you must define a logical name for that image in order to run an executable image linked against it. For example, if the file specification for STATSHR is SYS$SHARE:STATSHR.EXE, no logical name is necessary. But if you put STATSHR in SYS$DEVICE:[TEST], you must define STATSHR as a logical name before running an executable image that calls it. The logical name must be the same one that was used as the input file specification for the shareable image when it was linked (this is the same name used in installation). For example:

```
$ DEFINE STATSHR SYS$SYSDEVICE:[TEST]STATSHR
```

By redefining the logical name of a shareable image, you can replace that shareable image with another without requiring the calling executable image to relink. For example, the following statement redefines the file name STATSHR. It becomes the logical name of the shareable image SYS$SYSDEVICE:[MAIN]STATSHR.EXE for executable images calling STATSHR.

```
$ DEFINE STATSHR SYS$SYSDEVICE:[MAIN]STATSHR
```

Note

Logical names defined in the process or group logical name table are ignored when you run a privileged executable image. Only logical names and table names defined in executive or kernel modes are used to find the image.

Two shareable images installed with the /SHARED qualifier cannot have the same file name. (Use the INSTALL command REPLACE to update file versions.) For more information on the INSTALL command REPLACE, see the INSTALL section of the *OpenVMS System Management Utilities Reference Manual.*

16.9.14 Removing Known Images

The INSTALL command DELETE removes a known file list entry for an image and deletes all global sections created when the image was installed. Note the following restrictions on removing known images:

- A known image is not deleted as soon as the INSTALL DELETE command is entered. The deletion occurs only after all processes using the image have released it.
- A volume cannot be dismounted while any known file lists associated with it contain entries. To dismount a volume, you must delete all known images associated with it. You must also wait for all processes using those images to release them and for the system to write writable images back to their files. Use the DCL command SHOW DEVICES/FILES to determine the status of the files.

For more information on the INSTALL command DELETE, see the INSTALL section of the *OpenVMS System Management Utilities Reference Manual*.

17

Getting Information About the System

This chapter discusses setting up and maintaining system log files and using the Monitor utility.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Using the error logging facility	Section 17.3
Producing error log reports	Section 17.4
Setting up, managing, and printing the operator log file	Section 17.5
Using security auditing	Section 17.6
Using the Monitor utility to monitor system performance	Section 17.7

This chapter explains the following concepts:

Concept	Section
System log files	Section 17.1
Error logging facility	Section 17.2.1
Error Log utility	Section 17.2.2
Operator log file	Section 17.5.1
OPCOM messages	Section 17.5.2
Security auditing	Section 17.6.1
Monitor utility	Section 17.7.1

17.1 Understanding System Log Files

In maintaining your system, you need to collect and review information about system events. The operating system provides several log files that record information about the use of system resources, error conditions, and other system events. Table 17–1 briefly describes each file and provides references to sections that discuss the file in more detail.

Table 17–1 System Log Files

Log File	Description	For More Information
Error log file	The system automatically records device and CPU error messages in this file. The Error Log utility invokes the **Error Log Report Formatter (ERF)**, which selectively reports the contents of an error log file.	See Section 17.2.
Operator log file	The Operator Communication Manager (OPCOM) records system events in this file.	See Chapter 2, Section 17.5, and Section 18.6.
Accounting file	The accounting file tracks the use of system resources.	See Chapter 18.
Security audit log file	The audit server process writes security-relevant system events to this file.	See Section 17.6.

17.2 Understanding Error Logging

This section explains the error logging facility, which automatically writes error messages to the latest version of the error log file, SYS$ERRORLOG:ERRLOG.SYS. This section also describes the Error Log utility, which you can use to report selectively on error log files.

Error log reports are primarily intended for use by Digital Services personnel to identify hardware problems. System managers often find error log reports useful in identifying recurrent system failures that require outside attention.

17.2.1 Understanding the Error Logging Facility

The error logging facility consists of the three parts shown in Table 17–2.

Table 17–2 Parts of the Error Logging Facility

Part	Description
Executive routines	Detect errors and events and write relevant information into error log buffers in memory.
ERRFMT process	Periodically empties the error log buffers, transforms the descriptions of the errors into standard formats, and stores the formatted information in a file on the system disk. (The ERRFMT process starts when the system is booted.)
Error Log utility	Selectively reports the contents of an error log file; you invoke it by entering the DCL command ANALYZE/ERROR_LOG. (See Section 17.4.)

The executive routines and the ERRFMT process operate continuously without user intervention. The routines fill the error log buffers in memory with raw data on every detected error and event. When one of the available buffers becomes full, or when a time allotment expires, ERRFMT automatically writes the buffers to ERRLOG.SYS.

Sometimes a burst of errors can cause the buffer to fill up before ERRFMT can empty them. You can detect this condition by noting a skip in the error sequence number of the records reported in the error log reports. As soon as ERRFMT frees the buffer space, the executive routines resume preserving error information in the buffers.

The ERRFMT process displays an error message on the system console terminal and stops itself if it encounters excessive errors while writing the error log file. Section 17.3.1 explains how to restart the ERRFMT process.

17.2.2 Understanding the Error Log Utility (ERROR LOG)

The Error Log utility (ERROR LOG) is a system management tool that selectively reports the contents of one or more error log files. ERROR LOG supports most OpenVMS-supported hardware, such as adapters, disks, tapes, CPUs, and memories, but not all communications devices. Some synchronous communications devices are supported.

The operating system automatically writes messages to the latest version of an error log file named SYS$ERRORLOG:ERRLOG.SYS as the events shown in Table 17–3 occur.

Table 17–3 Types of Events Reported in the Error Log File

Event	Description
Errors	Device errors, device timeouts, machine checks, bus errors, memory errors (hard or soft error correcting code [ECC] errors), asynchronous write errors, and undefined interrupts
Volume changes	Volume mounts and dismounts
System events	System startups, messages from the Send Message to Error Logger ($SNDERR) system service, and time stamps

You can use the Error Log utility to process error log entries for the following forms of optional output:

- Full report of selected entries, which is the default
- Brief report of selected entries
- Summary report of selected entries
- Register dump report of selected device entries
- Binary copy of selected entries
- Binary copy of rejected entries

Section 17.4 explains how to produce error log reports. See the *OpenVMS System Management Utilities Reference Manual* for examples of error log reports.

The error reports that the Error Log utility produces are useful in two ways:

- They aid preventive maintenance by identifying areas within the system that show potential for failure.
- They aid the diagnosis of a failure by documenting the errors and events that led up to it.

The detailed contents of the reports are most meaningful to Digital Services personnel. However, you can use the reports as an important indicator of the system's reliability. For example, using the DCL command SHOW ERROR, you might see that a particular device is producing a relatively high number of errors. You can then use the Error Log utility to obtain a more detailed report and decide whether to consult Digital Services. If you do, Digital Services personnel can run diagnostic programs to investigate the device and attempt to isolate the source of the errors.

If a system component does fail, a Digital Services representative can study the error reports of the system activity leading up to and including the failure. If a device fails, you can generate error reports immediately after the failure; for example:

- One report might describe in detail all errors associated with the device that occurred within the last 24 hours.
- Another report might summarize all types of errors for all devices that occurred within the same time period.
- The summary report can put the device errors into a systemwide context.

The Digital Services representative can then run the appropriate diagnostic program for a thorough analysis of the failed device. Using the combined error logging and diagnostic information, the Digital Services representative can proceed to correct the device.

Error reports allow you to anticipate potential failures. Effective use of the Error Log utility in conjunction with diagnostic programs can significantly reduce the amount of system downtime.

17.3 Using the Error Logging Facility

The ERRFMT process is started automatically at boot time. This section explains how to restart the ERRFMT process, if necessary, and how to maintain error log files.

17.3.1 Restarting the ERRFMT Process

To restart the ERRFMT process (explained in Section 17.2.1), follow these steps:

1. Log in to the system manager's account so that you have the required privileges to perform the operation.
2. Execute the site-independent startup command procedure (STARTUP.COM), specifying ERRFMT as the command parameter, as follows:

   ```
   $ @SYS$SYSTEM:STARTUP ERRFMT
   ```

   Note

   If disk quotas are enabled on the system disk, ERRFMT starts only if UIC [1,4] has sufficient quotas.

17.3.2 Maintaining Error Log Files

Because the error log file, SYS$ERRORLOG:ERRLOG.SYS, is a shared file, ERRFMT can write new error log entries while the Error Log utility reads and reports on other entries in the same file.

ERRLOG.SYS increases in size and remains on the system disk until you explicitly rename or delete it. Therefore, you need to devise a plan for regular maintenance of the error log file. One method is to rename ERRLOG.SYS on a daily basis. If you do this, the system creates a new error log file. You might, for example, rename the current copy of ERRLOG.SYS to ERRLOG.OLD every morning at 9:00. To free space on the system disk, you can then back up the renamed version of the error log file on a different volume and delete the file from the system disk.

Another method is to keep the error log file on a disk other than the system disk by defining the logical name SYS$ERRORLOG to be the device and directory where you want to keep error log files; for example:

```
$ DEFINE/SYSTEM/EXECUTIVE DUA2:[ERRORLOG]
```

To define this logical name each time you start up the system, add the logical name definition to your SYLOGICALS.COM procedure. See Section 5.2.5 for details.

Be careful not to delete error log files inadvertently. You might also want to adopt a file-naming convention that includes a beginning or ending date for the data in the file name.

17.4 Producing Error Log Reports

You use the Error Log utility to report selectively on the contents of an error log file. You must have SYSPRV to run ERROR LOG.

You enter the DCL command in the following format:

ANALYZE/ERROR_LOG [/qualifier(s)][file_spec[,...]]

where:

qualifier	Specifies the function the ANALYZE/ERROR_LOG command is to perform.
file-spec	Specifies one or more files that contain information to be interpreted for the error log report.

See the *OpenVMS System Management Utilities Reference Manual* for details about the command and its parameters and for examples of error log reports.

ERROR LOG issues error messages for inconsistent error log entries. The *OpenVMS System Messages and Recovery Procedures Reference Manual* lists these messages and provides explanations and suggested user actions.

17.4.1 Producing a Full Error Log Report

The following steps show how to produce an error log report for all entries in the error log file and how to print the report:

1. Either log in to the SYSTEM account or ensure that you have the SYSPRV privilege. (You need this privilege to access the error log file.) For example:

   ```
   $ SET PROCESS/PRIVILEGE=SYSPRV
   ```

2. Set your default disk and directory to SYS$ERRORLOG:

   ```
   $ SET DEFAULT SYS$ERRORLOG
   ```

3. Examine the error log directory to see which error log file you want to analyze:

   ```
   $ DIRECTORY
   ```

4. To obtain a full report of the current error log file, enter the following command:

   ```
   $ ANALYZE/ERROR_LOG/OUTPUT=ERRORS.LIS
   ```

5. Print a copy of the report, using the file name you specified with the /OUTPUT qualifier:

   ```
   $ PRINT ERRORS.LIS
   ```

Example

```
  $ SET PROCESS/PRIVILEGE=SYSPRV
  $ SET DEFAULT SYS$ERRORLOG
❶ $ DIRECTORY
  Directory SYS$SYSROOT:[SYSERR]

  ERRLOG.OLD;2  ERRLOG.OLD;1  ERRLOG.SYS;1

  Total of 3 files.
❷ $ ANALYZE/ERROR_LOG/OUTPUT=ERRORS.LIS ERRLOG.OLD
❸ $ PRINT ERRORS.LIS
```

Following are explanations of the commands in the example.

❶ The DIRECTORY command lists all the files in the SYS$ERRORLOG directory. The directory contains three files: two old error log files and the current error log file, ERRLOG.SYS.

❷ The ANALYZE/ERROR_LOG command writes a full report to a file called ERRORS.LIS, using the most recent ERRLOG.OLD file as input.

❸ The PRINT command prints ERRORS.LIS.

17.4.2 Using Other Error Log Report Options

This section briefly explains how to specify report formats and produce a report of selected entries.

Table 17–4 contains error log report options. For more details about options and examples of error log reports using options, see the *OpenVMS System Management Utilities Reference Manual.*

Table 17–4 Error Log Report Options

In Order To...	You Can...
Specify report formats	Change report formats by using qualifiers, including the following: • /BINARY—to convert binary error log records to ASCII text or to copy error log records to a specified output file. • /BRIEF—to create a brief report. • /REGISTER_DUMP—to generate, in a hexadecimal longword format, a report that consists of device register information (used in conjunction with the /INCLUDE qualifier). • /REJECTED—to specify the name of a file that will contain binary records for rejected entries.
Specify a display device for reports	Use the /OUTPUT qualifier to send reports to a terminal for display or to a disk or magnetic tape file. By default, the system sends the report to the SYS$OUTPUT device. Because ERROR LOG reports are 72 columns wide, you can display them on the terminal screen.

(continued on next page)

Table 17–4 (Cont.) Error Log Report Options

In Order To...	You Can...
Produce a report of selected entries	Use qualifiers to produce error log reports for specific types of events and for a specified time interval. For example, you can process error log entries by selecting a time interval using the /SINCE, /BEFORE, or /ENTRY qualifiers. You can specify error log entries for specific events by using the qualifiers /INCLUDE and /EXCLUDE. These qualifiers form a filter to determine which error log entries are selected or rejected. In addition, you can generate error log reports for one or more VMScluster members by using the /NODE qualifier.
Exclude unknown error log entries	By default, when ANALYZE/ERROR_LOG encounters an unknown device, CPU, or error log entry, the utility produces the entry in hexadecimal longword format. Exclude these entries from the report by specifying /EXCLUDE=UNKNOWN_ENTRIES in the command line.

17.5 Setting Up, Maintaining, and Printing the Operator Log File

The following sections describe the contents of the operator log file and OPCOM messages. They also explain how to perform the following operations, which require OPER privilege:

Operation	For More Information
Set up the operator log file	Section 17.5.3
Manage the operator log file	Section 17.5.4
Print the operator log file	Section 17.5.5

17.5.1 Understanding the Operator Log File

The operator log file (SYS$MANAGER:OPERATOR.LOG) records system events and user requests that the Operator Communication Manager (OPCOM) sends to the operator terminal. This recording occurs even if all operator terminals have been disabled. By default, OPCOM starts when you boot your system. (For more information on OPCOM, see Section 2.4.)

You can use the operator log file to anticipate and prevent hardware and software failures and to monitor user requests for disk and magnetic tape operations. By regularly examining the operator log file, you can often detect potential problems and take corrective action.

17.5.2 Understanding OPCOM Messages

The following sections describe the types of messages that appear in the operator log file.

Type of Message	For More Information
Initialization messages	Section 17.5.2.1
Device status messages	Section 17.5.2.2

Type of Message	For More Information
Terminal enable and disable messages	Section 17.5.2.3
User request and operator reply messages	Section 17.5.2.4
Volume mount and dismount messages	Section 17.5.2.5
System parameter messages	Section 17.5.2.6
‡Security alarm messages	Section 17.5.2.7

‡AXP specific

Section 17.5.2.8 contains an example of typical kinds of messages found in an operator log file.

17.5.2.1 Initialization Messages

When you enter the REPLY/LOG command, the system closes the current operator log file and creates and opens a new version of the file. The system records all subsequent OPCOM messages in the new log file.

When you create a new log file, the first message recorded in it is an initialization message. This message shows the terminal name of the operator who initialized the log file, and the log file specification. This message appears in the following format:

```
%%%%%%%%%%%  %OPCOM, <dd-mmm-yyyy hh:mm:ss.cc> %%%%%%%%%%%
Logfile has been initialized by operator <terminal-name>
Logfile is <logfile-specification>
```

For example:

```
%%%%%%%%%%%  OPCOM, 19-APR-1993 12:29:24.52  %%%%%%%%%%%
Logfile has been initialized by operator _MARS$VTA2:
Logfile is HOMER::SYS$SYSMOND:[SYSMGT]OPERATOR.LOG;43
```

17.5.2.2 Device Status Messages

Some I/O drivers send messages to OPCOM concerning changes in the status of the devices they control. For example, when a line printer goes off line, an OPCOM message appears in the operator log file at periodic intervals until you explicitly return the device to online status.

The device status message appears in the operator log file in the following format:

```
%%%%%%%%%%%%  OPCOM <dd-mmm-yyyy hh:mm:ss.cc> %%%%%%%%%%%%
Device <device-name> is offline
```

This message can appear for card readers, line printers, and magnetic tapes.

17.5.2.3 Terminal Enable and Disable Messages

Following are examples of commands you can give to enable and disable terminals as operator terminals (or consoles) and explanations of the corresponding messages that appear in the operator log file.

REPLY/ENABLE Messages

To designate a terminal as an operator terminal, enter the REPLY/ENABLE command from the desired terminal. OPCOM confirms the request by displaying messages in the following format at the operator terminal and in the operator log file:

```
%%%%%%%%%%%%  %OPCOM dd-mmm-yyyy hh:mm:ss.cc  %%%%%%%%%%%%
Operator <terminal-name> has been enabled, username <user-name>

%%%%%%%%%%%%  %OPCOM dd-mmm-yyyy hh:mm:ss.cc  %%%%%%%%%%%%
Operator status for operator <terminal-name>
<status-report>
```

These messages tell you which terminal has been established as an operator terminal and lists the requests the terminal can receive and respond to.

You can also designate a terminal as an operator terminal for a particular function by entering the REPLY/ENABLE=*class* command.

If you enter the command REPLY/ENABLE=TAPES, for example, OPCOM displays messages similar to the following:

```
%%%%%%%%%%%%  %OPCOM 19-APR-1993 10:25:35.74 %%%%%%%%%%%%
Operator _ROUND$OPA1: has been enabled, username SYSTEM

%%%%%%%%%%%%  %OPCOM 19-APR-1993 10:25:38.82 %%%%%%%%%%%%
Operator status for operator _ROUND$OPA1:
CENTRAL, PRINTER, TAPES, DISKS, DEVICES, NETWORK, CLUSTER,
LICENSE, OPER1, OPER2
```

OPCOM confirms that the terminal is established as an operator terminal and indicates that the terminal can only receive and respond to requests concerning magnetic-tape-oriented events, such as the mounting and dismounting of tapes.

REPLY/DISABLE Messages

A terminal that you designate as an operator terminal automatically returns to nonoperator status when the operator logs out. To return the terminal to normal (nonoperator) status without logging out, enter the REPLY/DISABLE command from the terminal.

OPCOM confirms that the terminal is no longer an operator terminal by displaying a message both at the operator terminal and in the operator log file. The message, which tells you which terminal has been restored to nonoperator status and when the transition occurred, has the following format:

```
%%%%%%%%%%%  %OPCOM <dd-mmm-yyyy hh:mm:ss.cc> %%%%%%%%%%%
Operator <terminal-name> has been disabled, username <user-name>
```

If you designate a terminal as an operator terminal and only partial operator status is disabled, OPCOM displays a status message. This message lists which requests the terminal can still receive and respond to. This message is displayed at the operator terminal and in the operator log file in the following format:

```
%%%%%%%%%%%  %OPCOM  <dd-mmm-yyyy hh:mm:ss.cc>  %%%%%%%%%%%
Operator status for operator <terminal-name>
<status-report>
```

For example, suppose you designate a terminal as an operator terminal that receives messages concerning magnetic tapes and disks, as well as messages intended for the special site-specific operator class known as OPER10. Later, you relinquish the terminal's ability to receive messages concerning tapes. When you enter the REPLY/DISABLE=TAPES command, OPCOM returns a message like the following:

```
%%%%%%%%%%%  %Opcom 19-APR-1993 09:23:45.32  %%%%%%%%%%%
Operator status for operator TTA3
DISKS, OPER10
```

This message tells you that terminal TTA3 still receives and can respond to messages about disks and messages directed to OPER10.

17.5.2.4 User Request and Operator Reply Messages

To communicate with the operator, the user enters the REQUEST command, specifying either the /REPLY or /TO qualifier. Following are explanations of these qualifiers:

Request	Explanation

REQUEST /REPLY	If the user enters this command, the request is recorded in the operator log file in the following format:
	`%%%%%%%%%%%  %OPCOM <dd-mmm-yyyy hh:mm:ss.cc> %%%%%%%%%%%`
	`Request <request-id>, from user <user-name> on <node-name>`
	`<_terminal-name:>, <"message-text">`
	This message tells you which user sent the message, the time the message was sent, the request identification number assigned to the message, the originating terminal, and the message itself.

REQUEST /TO	If the user enters this command, the request is recorded in the operator log file in the format shown in the REQUEST/REPLY example, but without a request identification number:
	`%%%%%%%%%%%  %OPCOM, <dd-mmm-yyyy hh:mm:ss.cc> %%%%%%%%%%%`
	`Request from user <user-name> on <node-name>`
	`<_terminal-name:>, <"message-text">`

Messages also differ depending on how you reply to a user:

Reply	Explanation

REPLY/TO	When you respond to a user's request and specify the /TO qualifier, the response is recorded in the operator log file in the following format:
	`response message`
	`<hh:mm:ss.cc>, request <request-id> completed`
	`     by operator <terminal-name>`
	This message indicates how the operator responded to the user's request, as well as when the response was entered and which operator responded.

REPLY /ABORT	When you respond to a user's request and specify the /ABORT qualifier, the response is recorded in the operator log file in the following format:
	`<hh:mm:ss.cc>, request <request-id> was aborted`
	`     by operator <terminal-name>`

REPLY /PENDING	When you respond to a user's request using the /PENDING qualifier, the response is not recorded in the operator log file because the request has not yet been completed (that is, the request has not been fulfilled or aborted).

When a user enters a REQUEST/REPLY command and you have disabled all terminals as operators' terminals, OPCOM records all subsequent users' requests in the log file, but returns a message to the user indicating that no operator coverage is available.

All other OPCOM responses to REPLY commands, except responses involving the REPLY/ENABLE, REPLY/DISABLE, and REPLY/LOG commands, are not logged in the operator log file.

17.5.2.5 Volume Mount and Dismount Messages

Perhaps the widest range of operator messages occurs with volume mounts and dismounts; for example:

```
%%%%%%%%%%%  OPCOM, 19-APR-1993 22:41:07.54  %%%%%%%%%%%
message from user SYSTEM
Volume "KLATU      " dismounted, on physical device MTA0:
15-APR-1993 22:42:14.81, request 2 completed by operator OPA0
```

17.5.2.6 System Parameter Messages

Users with the appropriate privileges can change the following sets of values for system parameters:

Values	Description
Current	Values stored in the default parameter file on disk and used to boot the system
Active	Values stored in memory and used while the system is running

When the system boots, it reads the current values into memory, creating active values. An active value remains equal to the current value until you change either value.

Users can make the following changes to active and current system parameters:

- Active system parameters—Users with CMKRNL privilege can use the System Management utility (SYSMAN) or the System Generation utility (SYSGEN) to change system parameters in the running (active) system. Users can change only those active values that are categorized as *dynamic* system parameters.
- Current system parameters—Users with SYSPRV privilege can use SYSMAN or SYSGEN to change system parameters in the current system.

Note

Digital recommends that you use AUTOGEN or SYSMAN, not SYSGEN, to change system parameters, as explained in Section 14.2.

OPCOM logs all changes made to active and current system parameters with messages in the following format:

```
%%%%%%%%%%%  %OPCOM <dd-mmm-yyyy hh:mm:ss.cc> %%%%%%%%%%%
Message from user <user-name>
%SYSGEN-I-WRITExxx, <system-mode> system parameters modified by process ID
<process-id> into file <file-spec>
```

For example:

```
%%%%%%%%%%%  %OPCOM 3-AUG-1993 08:11:59.55 %%%%%%%%%%%
Message from user D_PLUTO on ANASAT
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process ID 0000020B
into file SYS$UPDATE:[SYSTEM]UPDATESYS.PAR;2
```

This message indicates that current system parameters have been changed.

Note

If you have changed the format of system messages with the DCL command SET MESSAGE, these messages might not appear in the log file.

17.5.2.7 Security Alarm Messages

Alarm messages are sent to the security operator terminal when selected events occur. See Section 17.6.6 for instructions.

AXP

On AXP systems, security alarm messages also appear in the operator log file if you enable a security operator terminal and specify alarm events. ♦

Following is an example of a security alarm OPCOM message:

```
%OPCOM, 19-APR-1993 12:27:52.26, security alarm on node HERA
System UAF record modification
```

17.5.2.8 Contents of an Operator Log File

Example 17–1 illustrates some typical messages found in an operator log file.

Example 17–1 Sample Operator Log File (SYS$MANAGER:OPERATOR.LOG)

```
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:26:07.90  %%%%%%%%%%%
❶ Device DMA0: is offline.
  Mount verification in progress.
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:26:20.22  %%%%%%%%%%%
  Mount verification completed for device DMA0:
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:33:54.07  %%%%%%%%%%%
❷ Operator '_ZEUS$VT333:' has been disabled, user JONES
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:34:15.47  %%%%%%%%%%%
  Operator '_ZEUS$VT333:' has been enabled, user SMITH
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:34:15.57  %%%%%%%%%%%
  operator status for '_ZEUS$VT333:'
  PRINTER, TAPES, DISKS, DEVICES
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:38:53.21  %%%%%%%%%%%
❸ request 1, from user PUBLIC
  Please mount volume KLATU in device MTA0:
  The tape is in cabinet A
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:39:54.37  %%%%%%%%%%%
  request 1 was satisfied.
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:40:23.54  %%%%%%%%%%%
❹ message from user SYSTEM
  Volume "KLATU       " mounted, on physical device MTA0:
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:40:38.02  %%%%%%%%%%%
  request 2, from user PUBLIC
  MOUNT new relative volume 2 () on MTA0:
  %%%%%%%%%%%  OPCOM, 19-APR-1993 22:41:07.54  %%%%%%%%%%%
  message from user SYSTEM
  Volume "KLATU       " dismounted, on physical device MTA0:
  15-APR-1993 22:42:14.81, request 2 completed by operator OPA0
```

(continued on next page)

Example 17–1 (Cont.) Sample Operator Log File (SYS$MANAGER:OPERATOR.LOG)

```
%%%%%%%%%%%  OPCOM, 19-APR-1993 22:46:47.96  %%%%%%%%%%%
request 4, from user PUBLIC
_TTB5:, This is a sample user request with reply expected.
%%%%%%%%%%%  OPCOM, 19-APR-1993 22:47:38.50  %%%%%%%%%%%
request 4 was canceled
%%%%%%%%%%%  OPCOM, 19-APR-1993 22:48:21.15  %%%%%%%%%%%
message from user PUBLIC
_TTB5:, This is a sample user request without a reply expected.
%%%%%%%%%%%  OPCOM, 19-APR-1993 22:49:37.64  %%%%%%%%%%%
Device DMA0: has been write locked.
Mount verification in progress.
%%%%%%%%%%%  OPCOM, 19-APR-1993 23:33:54.07  %%%%%%%%%%%
message from user NETACP
DECnet shutting down
```

The following messages appear in the example:

❶ Device status message (see Section 17.5.2.2)

❷ Terminal enable and disable message (see Section 17.5.2.3)

❸ User request and operator reply messages (see Section 17.5.2.4)

❹ Volume mount and dismount messages (see Section 17.5.2.5)

17.5.3 Setting Up the Operator Log File

The operator log file normally resides on the system disk in the [SYSMGR] directory. You can, however, maintain the log file in a different location by defining the logical name OPC$LOGFILE_NAME.

Because this file is in ASCII format, you can print it. Print copies regularly and retain these copies for reference. Section 17.5.5 describes how to print copies of the operator log file.

The system creates a new version of OPERATOR.LOG each time the system is rebooted (except on workstations in a VMScluster environment, where the log file is not opened by default). Note that there is one operator log file per node; it is not a shared file.

17.5.3.1 Creating a New Version of the Operator Log File

You can use the DCL command REPLY/LOG to create a new version of the file at any time. The highest version is always the one in use and is inaccessible to other users. By default, messages of all operator classes are in the log file.

Following are guidelines for using the REPLY/LOG command:

- You can use the REPLY/LOG/ENABLE=*(list-of-classes)* and REPLY/LOG /DISABLE=*(list-of-classes)* commands to specify which operator classes to include in the log file.
- When you use the /LOG qualifier with the REPLY/ENABLE and REPLY /DISABLE commands, the classes you select are enabled or disabled for the log file rather than for the terminal.

For more information, see the REPLY/LOG, REPLY/ENABLE, and REPLY /DISABLE commands in the *OpenVMS DCL Dictionary*.

Example

The following command opens a log file to include messages about mounting and dismounting disks and tapes:

```
$ REPLY/LOG/ENABLE=(DISKS,TAPES)
```

17.5.3.2 Specifying Logical Names

You can specify the default state of the operator log files by defining logical names in the command procedure SYS$MANAGER:SYLOGICALS.COM. The following table lists these logical names and their functions. For more information on SYLOGICALS.COM, see Section 5.2.5.

Logical Name	Function
OPC$LOGFILE_ ENABLE	Specifies whether an operator log file is opened. If defined to be true, an operator log file is opened. If defined to be false, no operator log file is opened. By default, a log file is opened on all systems except workstations in a VMScluster environment.
OPC$LOGFILE_ CLASSES	Specifies the operator classes that are enabled for the log file. By default, a log file is opened for all classes. The logical name can be a search list of the allowed classes, a comma-separated list, or a combination of the two. Note that you can define OPC$LOGFILE_ CLASSES even if you do not define OPC$LOGFILE_ENABLE. In that case, the classes are used for any log files that are opened, but the default is used to determine whether to open the log file.
OPC$LOGFILE_ NAME	Specifies the name of the log file. By default, the log file is named SYS$MANAGER:OPERATOR.LOG. If you specify a disk other than the system disk, include commands to mount that disk in the command procedure SYLOGICALS.COM.

Note

The only logical that is used for more than the initial startup of OPCOM is OPC$LOGFILE_NAME. All other OPCOM logicals are ignored. For example, a REPLY/LOG command opens a new operator log file even if the logical OPC$LOGFILE_ENABLE is defined to be false. To reset OPCOM states and classes after startup, use REPLY/ENABLE or REPLY /DISABLE commands.

17.5.3.3 Disabling Security Class Messages (AXP Only)

AXP

On AXP systems, by default OPCOM logs all SECURITY class messages in the operator log file. However, these entries duplicate the entries that the audit server process (AUDIT_SERVER) writes to the system security audit log. To conserve disk space, you can disable the SECURITY class in the operator log file.

Example

The following example shows how to define the logicals to disable SECURITY class messages in the operator log file:

```
$ DEFINE/SYSTEM OPC$LOGFILE_CLASSES CENTRAL,PRINTER,TAPES,DISKS, -
_$ DEVICES,NETWORK,CLUSTER,LICENSE,OPER1,OPER2,OPER3,OPER4,OPER5, -
_$ OPER6,OPER7,OPER8,OPER9,OPER10,OPER11,OPER12  ♦
```

17.5.4 Managing the Operator Log File

Devise a plan for regular maintenance of operator log files. One way is to start a new log file and rename the second-highest version daily. (See the example in the next section.) You might want to purge outdated versions of the operator log file on a regular basis. However, do not delete versions that you have not backed up. For more information, see Section 5.2.7.9.

If OPCOM is inadvertently deleted, follow these steps to start it manually:

1. Log in to the SYSTEM account so that you have the required privileges to perform the operation.
2. Enter the following command to execute the startup command procedure (STARTUP.COM), specifying OPCOM as the command parameter:

   ```
   $ @SYS$SYSTEM:STARTUP OPCOM
   ```

17.5.5 Printing the Operator Log File

Perform the following operation to produce a printed copy of the most recent version of the operator log file. (You must have OPER privilege.)

1. Use the following command to enable the terminal as an operator terminal:

   ```
   $ REPLY/ENABLE
   ```

2. Close the current log file and open a new one by entering the following command:

   ```
   $ REPLY/LOG
   ```

3. Set the default to SYS$MANAGER and enter the following command to list all versions of the file:

   ```
   $ SET DEFAULT SYS$MANAGER
   $ DIRECTORY OPERATOR.LOG
   ```

4. Rename the second-highest version to OPERATOR.OLD:

   ```
   $ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
   ```

   The version number, –1, specifies that you want to rename the second-highest version of this file. (The highest version number is the current operator log file.)

5. Print the operator log file by entering the following command:

   ```
   $ PRINT OPERATOR.OLD
   ```

Example

```
❶ $ REPLY/ENABLE
❷ $ REPLY/LOG

  %%%%%%%%%%%  OPCOM, 19-APR-1993 12:29:24.52  %%%%%%%%%%%
❸ Logfile has been initialized by operator _MARS$VTA2:
  Logfile is HOMER::SYS$MANAGER:[SYSMGT]OPERATOR.LOG

❹ $ SET DEFAULT SYS$MANAGER
❺ $ DIRECTORY OPERATOR.LOG

  Directory SYS$MANAGER:[SYSMGT]

  OPERATOR.LOG;582          OPERATOR.LOG;581

  Total of 2 files.
```

```
❻ $ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
❼ $ PRINT OPERATOR.OLD
```

Following are explanations of the numbered commands and responses in this example:

❶ The REPLY/ENABLE command enables the terminal as an operator terminal.

❷ The REPLY/LOG command closes the current log file and opens a new one.

❸ The response from OPCOM verifies that it has opened a new log file.

❹ The SET DEFAULT command sets the operator default disk to the system disk.

❺ The DIRECTORY command displays the files in the directory [SYSMGT] on the system disk.

❻ The RENAME command renames the second-highest version of the operator log file to OPERATOR.OLD.

❼ The PRINT command prints the old operator log file, OPERATOR.OLD.

17.6 Using Security Auditing

This section discusses how security auditing works; it also explains how to enable security auditing and how to create a new version of the security audit log file. For more information about the security audit log file, see the *Security Guide*.

17.6.1 Understanding Security Auditing

Security auditing is the act of recording security-relevant events as they occur on the system. Security-relevant events are divided into a number of categories called **event classes**.

By default, the system enables security auditing when you install or upgrade your system for the events shown in Table 17–5.

Table 17–5 Event Classes Audited by Default

Class	Description
†ACL	Access to any object holding a security Auditing ACE.
Audit	All uses of the SET AUDIT command. You cannot disable this category.
Authorization	All changes to the authorization database: • System user authorization file (SYSUAF) • Network proxy authorization file (NETPROXY) • Rights database (RIGHTSLIST)
Breakin	All break-in attempts: batch, detached, dialup, local, network, remote.
†Logfailure	All login failures: batch, dialup, local, remote, network, subprocess, detached.

†VAX specific

If the security requirements at your site justify additional auditing, you can enable security auditing for other event classes by using the DCL command SET AUDIT, as explained in Section 17.6.4.

The Security Audit Log File

The audit server process (created at system startup) records the events shown in Table 17–5 in the security audit log file.

VAX

On VAX systems, the security audit log file is called SYS$MANAGER:SECURITY.AUDIT$JOURNAL. ♦

AXP

On AXP systems, the security audit log file is called SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL. ♦

On AXP and VAX systems, the usefulness of the security audit log file depends upon the procedures you adopt to review the file on a regular basis. For example, you might implement the following procedure as part of your site audit review policy:

1. Create a new version of the security audit log file each morning.
2. Review the previous version of the log file for suspicious system activity. Depending on the number of security events that you are auditing on your system, it might be impractical to review every audit record written to the audit log file. In that case, you might want to select a specific set of records from the log file (for example, all Authorization and Breakin records, or all events created outside normal business hours).
3. If, during your review, you find any security events that appear suspicious, perform a more detailed inspection of the security audit log file, as described in the *Security Guide*.

17.6.2 Displaying Security Auditing Information

To see which event classes your site currently audits, you can enter the DCL command SHOW AUDIT.

VAX

Following is an example of security information displayed on VAX systems:

```
$ SHOW AUDIT

System security alarms currently enabled for:
  ACL
  Breakin:       dialup,local,remote,network,detached
  Privilege use:
    SECURITY
  Privilege failure:
    SECURITY
```

```
System security audits currently enabled for:
  ACL
  Authorization
  Breakin:       dialup,local,remote,network,detached
  Login:         dialup,local,remote,network,detached
  Logfailure:    batch,dialup,local,remote,network,subprocess,detached
  Logout:        dialup,local,remote,network,detached
  Privilege use:
    SECURITY
  Privilege failure:
    ACNT     ALLSPOOL ALTPRI   AUDIT    BUGCHK   BYPASS   CMEXEC   CMKRNL
    DETACH   DIAGNOSE EXQUOTA  GROUP    GRPNAM   GRPPRV   LOG_IO   MOUNT
    NETMBX   OPER     PFNMAP   PHY_IO   PRMCEB   PRMGBL   PRMMBX   PSWAPM
    READALL  SECURITY SETPRV   SHARE    SHMEM    SYSGBL   SYSLCK   SYSNAM
    SYSPRV   TMPMBX   VOLPRO   WORLD
  DEVICE access:
    Failure:     read,write,physical,logical,control
  FILE access:
    Failure:     read,write,execute,delete,control
  VOLUME access:
    Failure:     read,write,create,delete,control
```

♦

AXP

Following is an example of security information displayed on AXP systems:

```
$ SHOW AUDIT

Security alarm failure mode is set to:
    WAIT        Processes will wait for resource

Security alarms currently enabled for:
    ACL
    AUTHORIZATION
    BREAKIN:    (DIALUP,LOCAL,REMOTE,NETWORK,DETACHED)
    LOGIN:      (DIALUP,LOCAL,REMOTE,NETWORK,DETACHED)
    LOGOUT:     (DIALUP,LOCAL,REMOTE,NETWORK,DETACHED)
    FILE_ACCESS:
        FAILURE:   (READ,WRITE,EXECUTE,DELETE,CONTROL)
```

♦

17.6.3 Delaying Startup of Auditing

Ordinarily, the system turns on auditing in VMS$LPBEGIN just before SYSTARTUP_VMS.COM executes. You can change this behavior, however, by redefining the logical name SYS$AUDIT_SERVER_INHIBIT.

To change the point at which the operating system begins to deliver security-event messages, add the following line to the SYS$STARTUP:SYLOGICALS.COM command procedure:

```
$ DEFINE/SYSTEM/EXECUTIVE SYS$AUDIT_SERVER_INHIBIT YES
```

Then you can initiate auditing during another phase of system startup, perhaps at the end of SYSTARTUP_VMS.COM, by editing the command file to add the following line:

```
$ SET AUDIT/SERVER=INITIATE
```

For information on editing SYSTARTUP_VMS.COM, see Section 5.2.7.

17.6.4 Enabling Security Auditing for Additional Classes

To enable security auditing for classes in addition to those shown in Table 17–5, use the following format:

SET AUDIT/ENABLE=event-class[,...] {/ALARM | /AUDIT}

The *Security Guide* contains descriptions of event classes that you can enable.

When you enable auditing for additional event classes, you must specify two qualifiers: the /ENABLE qualifier and either /ALARM or /AUDIT.

AXP

On AXP systems, the /ALARM qualifier sends security event messages to all enabled security terminals and to the security audit log file. If the log file is enabled for security class messages, the formatted alarms are also written to the operator log file.

Because these messages duplicate the more compact system security audit log file, Digital recommends that you disable the security class messages for the operator log file. ♦

VAX

On VAX systems, you can use both /ALARM and /AUDIT. ♦

Following are explanations of the /ENABLE, /ALARM, and /AUDIT qualifiers.

Qualifier	Explanation
/ENABLE	Defines which event classes you want audited. See Chapter 18 for more information.
/ALARM /AUDIT	Defines the destination of the event message. • /ALARM directs the message to all enabled security operator terminals. • /AUDIT directs the message to the security audit log file. Use the /ALARM and /AUDIT qualifiers to report critical events. Less critical events can be written only to the security audit log file for later examination. The default event classes listed in Table 17–5 are sent as both alarms and audits.

The system begins auditing new events on all nodes as soon as you enable them.

Examples

The command in the following example enables auditing for volume mounts and dismounts.

```
$ SET AUDIT/ENABLE=MOUNT
```

VAX

The command in the following example for VAX systems enables auditing of unsuccessful file accesses.

```
$ SET AUDIT/AUDIT/ENABLE=ACCESS=FAILURE/CLASS=FILE
```
♦

17.6.5 Disabling Security Auditing

The system continues auditing until you explicitly disable the classes with the /DISABLE qualifier using the following syntax:

SET AUDIT/DISABLE=event-class[,...] {/ALARM | /AUDIT}

17.6.6 Enabling a Terminal to Receive Alarm Messages

The system sends alarm messages to terminals enabled for security class messages. In most cases, these security alarms appear on the system console by default. Since messages scroll quickly off the screen, it is good practice to enable a separate terminal for security class messages and disable message delivery to the system console.

Either choose a terminal in a secure location that provides hardcopy output, or have dedicated staff to monitor the security operator terminal. You can enable any number of terminals as security operators.

To set up a terminal to receive security class alarms, enter the following DCL command from the designated terminal:

```
$ REPLY/ENABLE=SECURITY
```

VAX

On VAX systems, security alarm messages are not written to the operator log file. They appear only on terminals enabled for security class messages.

The following example shows a security alarm message on a VAX system:

```
%%%%%%%%%%%  OPCOM  25-JUL-1993 16:07:09.20  %%%%%%%%%%%
Message from user AUDIT$SERVER on GILMORE
Security alarm (SECURITY) on GILMORE, system id: 20300
Auditable event:          Process suspended ($SUSPND)
Event time:               25-JUL-1993 16:07:08.77
PID:                      30C00119
Process name:             Hobbit
Username:                 HUBERT
Process owner:            [LEGAL,HUBERT]
Terminal name:            RTA1:
Image name:               $99$DUA0:[SYS0.SYSCOMMON.][SYSEXE]SET.EXE
Status:                   %SYSTEM-S-NORMAL, normal successful completion
Target PID:               30C00126
Target process name:      SMISERVER
Target username:          SYSTEM
Target process owner:     [SYSTEM]  ♦
```

AXP

The following example shows a security alarm message on an AXP system:

```
%%%%%%%%%%%   OPCOM   19-APR-1993 15:15:03.21   %%%%%%%%%%%
Message from user AUDIT$SERVER on LASSIE
Security alarm (SECURITY) and security audit (SECURITY) on LASSIE, system id:
19611
Auditable event:        Attempted file access
Event time:             19-APR-1993 15:15:03.20
PID:                    20200283
Username:               WILSON
Image name:             LASSIE$DMA0:[SYS0.SYSCOMMON.] [SYSEXE]DIRECTORY.EXE
Object name:            LASSIE$DMA0:[000000]000000.DIR;1
Object type:            file
Access requested:       READ
Status:                 %SYSTEM-F-NOPRIV, no privilege for attempted operation  ♦
```

17.6.7 Generating Security Reports

The most common type of report to generate is a brief, daily listing of events. You can create a command procedure that runs in a batch job every evening before midnight to generate a report of the day's security event messages and send it to the system manager via MAIL.

The following examples show the ANALYZE/AUDIT command line you would use to generate this type of report:

VAX

On VAX systems:

```
$ ANALYZE/AUDIT/SINCE=TODAY/OUTPUT=31DEC1993.AUDIT -
_$ SYS$MANAGER:SECURITY.AUDIT$JOURNAL
$ MAIL/SUBJECT="Security Events" 31DEC1993.AUDIT SYSTEM
```
♦

AXP

On AXP systems:

```
$ ANALYZE/AUDIT/SINCE=TODAY/OUTPUT=31DEC1993.AUDIT -
_$ SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL
$ MAIL/SUBJECT="Security Events" 31DEC1993.AUDIT SYSTEM
```
♦

17.6.8 Creating a New Version of the Security Audit Log File

Because the security audit log file continues to grow until you take action, you must devise a plan for maintaining it.

You use the following SET AUDIT command to create a new version of the clusterwide security audit log file. To prevent the loss of audit messages, the previous version of the audit log file is not closed until all audit messages stored in memory are written to the file.

17.6.8.1 Creating a New Clusterwide Version of the Log File

To open a new, clusterwide version of the security audit log file, use the following command:

```
$ SET AUDIT/SERVER=NEW_LOG
```

The audit server process opens a new version of the audit log file on each cluster node.

After you open the new log, rename the old version, using a naming convention for your files that incorporates in the file name a beginning or ending date for the data. Then copy the file to another disk, delete the log from the system disk to save space, and run the Audit Analysis utility on the old log.

By archiving this file, you maintain a clusterwide history of auditing messages. If you ever discover a security threat on the system, you can analyze the archived log files for a trail of suspicious user activity during a specified period of time.

17.6.8.2 Creating a New Node-Specific Version of the Log File

In some cases, VMScluster nodes might not share the same system security audit log file. To create a new, node-specific version of the security audit log file, use the following commands:

```
$ SET AUDIT/DESTINATION=filespec
$ SET AUDIT/SERVER=NEW_LOG
```

For the *filespec*, include a logical name that points to a node-specific file; for example, SYS$SPECIFIC:[SYSMGR]SECURITY. System security audit log files on other nodes are unaffected.

17.7 Monitoring Operating System Performance

The Monitor utility (MONITOR) is a system management tool that you can use to obtain information on operating system performance. You can specify MONITOR qualifiers to collect system performance data from the running system or play back data recorded previously in a recording file. When you play back data, you can display it, summarize it, and even rerecord it to reduce the amount of data in the recording file.

Following an explanation of the Monitor utility are sections that tell how to perform these operations:

Operation	For More Information
Invoke the Monitor utility	Section 17.7.2
Use live display monitoring	Section 17.7.3
Use live recording monitoring	Section 17.7.4
Use concurrent display and recording monitoring	Section 17.7.5
Use playback monitoring	Section 17.7.6
Use remote playback monitoring	Section 17.7.7
Rerecord monitoring	Section 17.7.8
Run MONITOR continuously	Section 17.7.9

For additional information about interpreting the information the Monitor utility provides, see the *Guide to OpenVMS Performance Management*. For additional information about using the Monitor utility, see the *OpenVMS System Management Utilities Reference Manual*.

17.7.1 Understanding the Monitor Utility (MONITOR)

Using MONITOR, you can monitor classes of systemwide performance data (such as system I/O statistics, page management statistics, and time spent in each of the processor modes) at specifiable intervals, and produce several types of output. You can also develop a database of performance information for your system by running MONITOR continuously as a background process, as explained in Section 17.7.9.

17.7.1.1 MONITOR Classes

Each MONITOR class consists of data items that, taken together, provide a statistical measure of a particular system performance category. The data items defined for individual classes are listed in the description of the MONITOR command in the *OpenVMS System Management Utilities Reference Manual*.

To monitor a particular class of information, you specify a class name on the MONITOR command line. The information MONITOR displays depends on the type of class you select. Table 17–6 compares the two MONITOR class types.

Table 17–6 Types of MONITOR Classes

Type of class	Description
System	Provides statistics on resource use for the entire system
Component	Provides statistics on the contribution of individual components to the overall system or VMScluster

As an example of the distinction between types of MONITOR classes, the IO class includes a data item to measure all direct I/O operations for the entire system, and is therefore a system class. The DISK class measures direct I/O operations for individual disks, and is therefore a component class.

Table 17–7 describes each MONITOR class and indicates whether it is a system or component class.

Table 17–7 MONITOR Classes

Class	Type	Description
ALL_CLASSES	System or Component	Statistics for all classes
CLUSTER	System	Clusterwide performance statistics
DECNET	System	DECnet for OpenVMS statistics
DISK	Component	Disk I/O statistics
DLOCK	System	Distributed lock management statistics
FCP	System	File control primitive statistics
FILE_SYSTEM_CACHE	System	File system cache statistics
IO	System	System I/O statistics
LOCK	System	Lock management statistics
MODES	Component	Time spent in each of the processor modes
MSCP_SERVER	System	MSCP server statistics
PAGE	System	Page management statistics
PROCESSES	Component	Statistics on all processes
RMS	Component	Record Management Services statistics
SCS	Component	System Communications Services statistics
STATES	System	Number of processes in each of the scheduler states
SYSTEM	System	Summary of statistics from other classes
TRANSACTION	System	DECdtm services statistics
†VBS	System	Virtual balance slot statistics
VECTOR	System	Vector processor scheduled usage

†VAX specific

17.7.1.2 Display Data

Except in the PROCESSES class, all displayable data items are rates or levels:

- **Rates** are shown in number of occurrences per second.
- **Levels** are values that indicate the size of the monitored data item.

You can request any or all of four different statistics for each data item:

Statistic	Description
Current rate or level	Most recently collected value for the rate or level
Average rate or level	Measured from the beginning of the MONITOR request
Minimum rate or level	Measured from the beginning of the MONITOR request
Maximum rate or level	Measured from the beginning of the MONITOR request

For the DISK, MODES, SCS, and STATES classes, you can optionally express all statistics as percentages.

In the PROCESSES class, MONITOR displays descriptive information, level information, and counters that increase over time.

17.7.1.3 Output Types

MONITOR collects system performance data by class and produces three forms of optional output, depending on the qualifier you specify:

Qualifier	Description
/DISPLAY	Produces output in the form of ASCII screen images, which are written at a frequency governed by the /VIEWING_TIME qualifier.
/RECORD	Produces a binary recording file containing data collected for requested classes; one record for each class is written per interval.
/SUMMARY	Produces an ASCII file containing summary statistics for all requested classes over the duration of the MONITOR request.

If you specify /INPUT with any of these qualifiers, MONITOR collects performance data from one or more previously created recording files; otherwise, data is collected from counters and data structures on the running system.

You use the /BEGINNING and /ENDING qualifiers to specify, respectively, when you want a MONITOR request to begin and end.

Using the /DISPLAY Qualifier

Information collected by MONITOR is normally displayed as ASCII screen images. You can use the optional /DISPLAY qualifier to specify a disk file to contain the information. If you omit the file specification, output is directed to SYS$OUTPUT.

Note

Be careful when you use the /DISPLAY qualifier. Because MONITOR enters display information into the file continuously, its size can grow very quickly.

See the *OpenVMS System Management Utilities Reference Manual* for a discussion of the /DISPLAY qualifier.

Using the /RECORD Qualifier

When you use the /RECORD qualifier, all data pertaining to the class is recorded, even if you are concurrently displaying only a single statistic or a single item of a component statistics class. The file is created when a MONITOR request is initiated and closed when a request terminates. You can use the resulting file as a source file for later requests to format and display the data on a terminal, to create a summary file, or to create a new recording file with different characteristics.

17.7.2 Invoking the Monitor Utility

To invoke the Monitor utility, enter the following DCL command:

```
$ MONITOR
```

MONITOR then displays the following prompt:

```
MONITOR>
```

In response to the prompt, you can enter any of the MONITOR commands, which are described in *OpenVMS System Management Utilities Reference Manual*. The most frequently used MONITOR command, however, specifies a class name.

Example

```
MONITOR> MONITOR PAGE
```

In this example, you specify the PAGE class name in the MONITOR command to monitor page management statistics.

You can also use the MONITOR command from DCL command level.

How to Override or Terminate a MONITOR Request

Generally, each MONITOR request runs until the time specified or implied by the /ENDING qualifier. However, to override or terminate a MONITOR request, you can press one of the following:

Keys	Description
Ctrl/W	Temporarily overrides a /VIEWING_TIME value and generates a new display immediately following the current one. This feature is useful when a broadcast message overwrites the MONITOR display area. You can also use Ctrl/W in conjunction with a large /VIEWING_TIME value to generate display events on demand.
Ctrl/C	Terminates the current request without exiting from the utility. You can then initiate a new request or enter any MONITOR command at the MONITOR> prompt.
Ctrl/Z	Terminates the current request and exits from MONITOR.

17.7.3 Using Live Display Monitoring

The following examples show how to use the live display monitoring mode of operation. Use this mode when you want to examine the activity of a running system, either on a routine basis or as part of an installation checkout, tuning, or troubleshooting exercise. The system does not keep a historical record of output.

Examples

1. `$ MONITOR PROCESSES/TOPCPU`

   The command displays a bar graph showing the eight processes that were the top consumers of CPU time during the period between displays. It also displays the amount of CPU time each process used.

   The command might produce a display similar to the following:

```
                        VAX/VMS Monitor Utility
                         TOP CPU TIME PROCESSES
                             on node BOMBAY
                          20-NOV-1993 10:06:49

                               0         25        50        75       100
                               + - - - - + - - - - + - - - - + - - - - -+
07E00181  CAFARET          100 ****************************************
                               |         |         |         |          |
                               |         |         |         |          |
                               |         |         |         |          |
                               |         |         |         |          |
                               |         |         |         |          |
                               |         |         |         |          |
                               + - - - - + - - - - + - - - - + - - - - -+
```

This example shows that user CAFARET is using 100 percent of the CPU time available. To display more information about the computer resources a user is using, use a command similar to the following:

```
$ SHOW PROCESS/CONTINUOUS/ID=07E00181
```

For this example, the most useful information in the resulting display is the name of the image at the end of the display; for example:

```
.
.
.
$1$DUA1:[SYS1D.SYSCOMMON.][SYSEXE]RODAN.EXE
```

This example indicates that CAFARET is running RODAN.EXE, which might be new software that is unintentionally running in a loop. This situation would occur if CAFARET were a privileged user running a process at a higher priority than other users.

2. `$ MONITOR/DISPLAY=PROCESSES.LOG PROCESSES`

   You can route MONITOR display output to any supported terminal device or to a disk file. This command writes MONITOR's display process statistics to the file PROCESSES.LOG. You can then print this file on a hardcopy device.

   Caution

   Because data is continuously added to the display file, be careful that the file does not grow too large.

3.
```
$ MY_CLASSES :== -
_$ "DECNET+FCP+IO+LOCK+MODES+PAGE+PROCESSES+STATES"
$ MONITOR/NODE=(CURLEY,LARRY)/INTERVAL=20/VIEWING_TIME=8 'MY_CLASSES'
```

   You might find it convenient to establish DCL symbols for frequently used combinations of class names, as in this example. The MONITOR command collects selected classes of data for VMScluster nodes CURLEY and LARRY every 20 seconds. Every 8 seconds, the system displays the most recently collected data for one of the classes. MONITOR predetermines the ordering of the classes for display.

17.7.4 Using Live Recording Monitoring

The following example shows how to use the live recording mode of operation. Use live recording when you need to capture MONITOR data for future use. Possible uses include the following:

- Installation checkout, tuning, troubleshooting; that is, all the uses that are listed for live display monitoring.

  Choose recording over display when you want to capture more classes than you can reasonably watch at a terminal, when a terminal is not available, or when you need to gather data about the system but cannot spend time at the terminal until later.

- Routine performance data gathering for long-term analysis.

  You can record MONITOR data on a routine basis and summarize it to gather data about system resource use over long periods of time.

Caution

Because data is continuously added to the recording file, be careful that the file does not grow too large.

Example

```
$ MONITOR/NODE=(LARRY,MOE)/NODISPLAY/RECORD MODES+STATES
```

The command in this example records data on the time spent in each processor mode and on the number of processes in each scheduler state for nodes LARRY and MOE. The command does not display this information.

17.7.5 Using Concurrent Display and Recording Monitoring

The following examples show how to use the concurrent display and recording mode of operation. Use this mode when you want to both retain performance data and watch as it is being collected. Because MONITOR allows shared read access to the recording file, a separate display process can play back the recording file as it is being written by another process.

The first example both collects and records data in the same command. The second and third examples show how you can perform concurrent recording and display using two separate processes: the process in the second example performs recording; the process in the third example plays back the file to obtain a summary.

Examples

1.
   ```
   $ MONITOR/RECORD FCP/AVERAGE,FILE_SYSTEM_CACHE/MINIMUM
   ```

   This command collects and records file system data and file system cache data every 3 seconds. It also displays, in bar graph form, average FCP statistics and minimum FILE_SYSTEM_CACHE statistics. The display alternates between the two graphs every 3 seconds. You can obtain current statistics in a subsequent playback request.

2.
   ```
   $ MONITOR/RECORD=SYS$MANAGER:ARCHIVE.DAT -
   _$ /INTERVAL=300/NODISPLAY ALL_CLASSES
   ```

   This command archives data for all classes once every 5 minutes. You might find it convenient to execute a similar command in a batch job, taking care to monitor disk space usage.

3.
```
$ MONITOR/INPUT=SYS$MANAGER:ARCHIVE.DAT: -
_$ /NODISPLAY/SUMMARY/BEGINNING="-1" PAGE,IO
```

The command in this example produces a summary of page and I/O activity that occurred during the previous hour, perhaps as part of an investigation of a reported performance problem. Note that because the recording process executes an OpenVMS RMS flush operation every 5 minutes, up to 5 minutes of the most recently collected data is not available to the display process.

You can specify the time between flush operations explicitly with the /FLUSH_INTERVAL qualifier. Note also that the display process must have read access to the recording file.

17.7.6 Using Playback Monitoring

The following examples show how to use the playback mode of operation. Use playback of a recording file to obtain terminal output and summary reports of all collected data or a subset of it. You can make a subset of data according to class, node, or time segment. For example, if you collect several classes of data for an entire day, you can examine or summarize the data on one or more classes during any time period in that day.

You can also display or summarize data with a different interval than the one at which it was recorded. You control the actual amount of time between displays of screen images with the /VIEWING_TIME qualifier.

Examples

1.
```
$ MONITOR/RECORD/INTERVAL=5 IO
    .
    .
    .
$ MONITOR/INPUT IO
```

The commands in this example produce system I/O statistics. The first command gathers and displays data every 5 seconds, beginning when you enter the command and ending when you press Ctrl/Z. In addition, the first command records binary data in the default output file MONITOR.DAT.

The second command plays back the I/O statistics display, using the data in MONITOR.DAT for input. The default viewing time for the playback is 3 seconds, but each screen display represents 5 seconds of monitored I/O statistics.

2.
```
$ MONITOR/RECORD/NODISPLAY -
_$ /BEGINNING=08:00:00 -
_$ /ENDING=16:00:00 -
_$ /INTERVAL=120 DISK
    .
    .
    .
$ MONITOR/INPUT/DISPLAY=HOURLY.LOG/INTERVAL=3600 DISK
```

The sequence of commands in this example illustrates data recording with a relatively small interval and data playback with a relatively large interval. This is useful for producing average, minimum, and maximum statistics that cover a wide range of time, but have greater precision than if they had been gathered using the larger interval.

The first command records data on I/O operations for all disks on the system for the indicated 8-hour period, using an interval of 2 minutes. The second command plays the data back with an interval of 1 hour, storing display output in the file HOURLY.LOG. You can then type or print this file to show the cumulative average disk use at each hour throughout the 8-hour period.

Note

The current statistic in HOURLY.LOG shows the current data in terms of the original collection interval of 120 seconds, not the new collection interval of 3600 seconds.

3. `$ MONITOR/INPUT/NODISPLAY/SUMMARY=DAILY.LOG DISK`

   The command in this example uses the recording file created in the previous example to produce a one-page summary report file showing statistics for the indicated 8-hour period. The summary report has the same format as a screen display. For example:

```
                         VAX/VMS Monitor Utility
                           DISK I/O STATISTICS
                              on node TLC          From: 25-NOV-1993 08:00:00
                                SUMMARY            To:   25-NOV-1993 16:00:00
I/O Operation Rate                          CUR        AVE        MIN        MAX

DSA0:                  SYSTEM_0            0.53       1.50       0.40       3.88
DSA1:                  SYSTEM_1            0.00       0.39       0.00       8.38
DSA4:                  WORK_0              0.00       0.11       0.00       1.29
DSA5:                  WORK_1              0.03       0.87       0.00       5.95
DSA6:                  WORK_2              0.03       0.25       0.00       2.69
DSA7:                  WORK_3              0.04       0.97       0.00      20.33
DSA17:                 TOM_DISK            0.00       0.04       0.00       0.80
DSA23:                 MKC                 0.00       0.00       0.00       0.13
$4$DUA0:      (RABBIT) SYSTEM_0            0.20       0.65       0.17       1.97
$4$DUA2:      (RABBIT) SYSTEM_0            0.20       0.65       0.17       1.97
$4$DUA3:      (RABBIT) SYSTEM_1            0.00       0.14       0.00       2.49

 PLAYBACK                            SUMMARIZING
```

17.7.7 Using Remote Playback Monitoring

If suitably privileged, you can collect MONITOR data from any system to which your system has a DECnet connection. You can then display the data live on your local system. To do so, follow these steps:

1. In the default DECnet directory on each remote system, create a file named MONITOR.COM, similar to the following:

```
$ !
$ !    * Enable MONITOR remote playback *
$ !
$ MONITOR /NODISPLAY/RECORD=SYS$NET ALL_CLASSES
```

2. On your local system, define a logical name for the remote system from which you want to collect data. Use the following syntax:

   DEFINE remotenodename_mon node::task=monitor

   You might want to define, in a login command procedure, a series of logical names for all the systems you want to access.

3. To display the remote MONITOR data as it is being collected, enter a command using the following syntax:

   MONITOR/INPUT=remotenodename_mon classnames

You can also place MONITOR.COM files in directories other than the default DECnet directory and use access control strings or proxy accounts to invoke these command files remotely.

When you invoke MONITOR on your local system, a process is created on the remote system that executes the MONITOR.COM command file. The remote system therefore experiences some associated CPU and DECnet overhead. You can regulate the overhead in the MONITOR.COM file by using the /INTERVAL qualifier and the list of class names.

17.7.8 Rerecording Monitoring

Rerecording is a combination of playback and recording. You can use it for data reduction of recording files. When you play back an existing recording file, all MONITOR options are available to you; thus, you can choose to record a subset of the recorded classes and a subset of the recorded time segment and a larger interval value.

All these techniques produce a new, smaller recording file at the expense of some of the recorded data. A larger interval value reduces the volume of the collected data, so displays and summary output produced from the newer recorded file will be less precise. Note that average rate values are not affected in this case, but average level values are less precise (since the sample size is reduced), as are maximum and minimum values.

Example

The following example shows how to use the rerecording mode of operation:

```
$ SUBMIT MONREC.COM
```

MONREC.COM contains the following commands:

```
$ MONITOR/NODISPLAY/RECORD/INTERVAL=60 /BEGINNING=8:00/ENDING=16:00 DECNET,LOCK
$ MONITOR/INPUT/NODISPLAY/RECORD DECNET
```

The first command runs in a batch job, recording DECnet and lock management data once every minute between the hours of 8 A.M. and 4 P.M. The second command, which is issued after the first command completes, rerecords the data by creating a new version of the MONITOR.DAT file, containing only the DECnet data.

17.7.9 Running MONITOR Continuously

You can develop a database of performance information for your system by running MONITOR continuously as a background process. This section contains examples of procedures that you, as cluster manager, might use to create multifile clusterwide summaries.

You can adapt the command procedures to suit conditions at your site. Note that you must define the logical names SYS$MONITOR and MON$ARCHIVE in SYSTARTUP.COM before executing any of the command files.

The directory with the logical name SYS$EXAMPLES includes three command procedures that you can use to establish the database. Instructions for installing and running the procedures are in the comments at the beginning of each procedure. Table 17–8 contains a brief summary of these procedures.

Table 17–8 MONITOR Command Procedures

Procedure	Description
MONITOR.COM	Creates a summary file from the recording file of the previous boot, and then begins recording for this boot. The recording interval is 10 minutes.
MONSUM.COM	Generates two clusterwide multifile summary reports that are mailed to the system manager: one report is for the previous 24 hours, and the other is for the previous day's prime-time period (9 A.M. to 6 P.M.). The procedure resubmits itself to run each day at midnight.
SUBMON.COM	Starts MONITOR.COM as a detached process. Invoke SUBMON.COM from the site-specific startup command procedure.

While MONITOR records data continuously, a summary report can cover any finite time segment. The MONSUM.COM command procedure, which is executed every midnight, produces and mails the two multifile summary reports described in Table 17–8. Because these reports are not saved as files, to keep them, you must either extract them from your mail file or alter the MONSUM.COM command procedure to save them.

17.7.9.1 Using the MONITOR.COM Procedure

The procedure in Example 17–2 archives the recording file and summary file from the previous boot and initiates continuous recording for the current boot. (Note that this procedure does not purge recording files.)

Example 17–2 MONITOR.COM Procedure

```
$ SET VERIFY
$ !
$ !  MONITOR.COM
$ !
$ !  This command file is to be placed in a cluster-accessible directory
$ !  called SYS$MONITOR and submitted at system startup time as a detached
$ !  process via SUBMON.COM.  For each node, MONITOR.COM creates, in
$ !  SYS$MONITOR, a MONITOR recording file that is updated throughout the
$ !  life of the boot. It also creates, in MON$ARCHIVE, a summary file from
$ !  the recording file of the previous boot, along with a copy of that
$ !  recording file.  Include logical name definitions for both cluster-
$ !  accessible directories, SYS$MONITOR and MON$ARCHIVE, in SYSTARTUP.COM.
$ !
$ SET DEF SYS$MONITOR
$ SET NOON
$ PURGE MONITOR.LOG/KEEP:2
$ !
$ !  Compute executing node name and recording and summary file names
$ !  (incorporating node name and date).
$ !
$ NODE = F$GETSYI("NODENAME")
$ SEP = ""
$ IF NODE .NES. "" THEN SEP = "_"
$ DAY = F$EXTRACT (0,2,F$TIME())
$ IF F$EXTRACT(0,1,DAY) .EQS. " " THEN DAY = F$EXTRACT(1,1,DAY)
$ MONTH = F$EXTRACT(3,3,F$TIME())
$ ARCHFILNAM = "MON$ARCHIVE:"+NODE+SEP+"MON"+DAY+MONTH
$ RECFIL = NODE+SEP+"MON.DAT"
$ SUMFIL = ARCHFILNAM+".SUM"
```

(continued on next page)

Example 17–2 (Cont.) MONITOR.COM Procedure

```
$ !
$ !  Check for existence of recording file from previous boot and skip
$ !  summary if not present.
$ !
$ OPEN/READ/ERROR=NORECFIL RECORDING 'RECFIL'
$ CLOSE RECORDING
$ !
$ !
$ !  Generate summary file from previous boot.
$ !
$ MONITOR /INPUT='RECFIL' /NODISPLAY /SUMMARY='SUMFIL' -
$ ALL_CLASSES+MODE/ALL+STATES/ALL+SCS/ITEM=ALL+SYSTEM/ALL+DISK/ITEM=ALL
$ !
$ !
$ !  Compute subject string and mail summary file to cluster manager.
$ !
$ !
$ A="""
$ B=" MONITOR Summary "
$ SUB = A+NODE+B+F$TIME()+A
$ MAIL/SUBJECT='SUB' 'SUMFIL' CLUSTER_MANAGER
$ !
$ !
$ !  Archive recording file and delete it from SYS$MONITOR.
$ !
$ COPY 'RECFIL' 'ARCHFILNAM'.DAT
$ DELETE 'RECFIL';*
$ !
$ NORECFIL:
$ SET PROCESS/PRIORITY=15
$ !
$ !
$ !  Begin recording for this boot.  The specified /INTERVAL value is
$ !  adequate for long-term summaries; you might need a smaller value
$ !  to get reasonable "semi-live" playback summaries (at the expense
$ !  of more disk space for the recording file).
$ !
$ MONITOR /INTERVAL=300 /NODISPLAY /RECORD='RECFIL' ALL_CLASSES
$ !
$ !
$ !  End of MONITOR.COM
$ !
```

17.7.9.2 Using the SUBMON.COM Procedure

The procedure in Example 17–3 submits MONITOR.COM as a detached process from SYSTARTUP.COM to initiate continuous recording for the current boot.

Example 17–3 SUBMON.COM Procedure

```
$ SET VERIFY
$ !
$ !  SUBMON.COM
```

(continued on next page)

Example 17–3 (Cont.) SUBMON.COM Procedure

```
$ !
$ !  This command file is to be placed in a cluster-accessible directory
$ !  called SYS$MONITOR.  At system startup time, for each node, it is
$ !  executed by SYSTARTUP.COM, following logical name definitions for
$ !  the cluster-accessible directories SYS$MONITOR and MON$ARCHIVE.
$ !
$ !
$ !  Submit detached MONITOR process to do continuous recording.
$ !
$ !
$ RUN   SYS$SYSTEM:LOGINOUT.EXE -
        /UIC=[1,4]              -
        /INPUT=SYS$MONITOR:MONITOR.COM  -
        /OUTPUT=SYS$MONITOR:MONITOR.LOG -
        /ERROR=SYS$MONITOR:MONITOR.LOG  -
        /PROCESS_NAME="Monitor" -
        /WORKING_SET=512 -
        /MAXIMUM_WORKING_SET=512 -
        /EXTENT=512/NOSWAPPING
$ !
$ !
$ !  End of SUBMON.COM
$ !
```

17.7.9.3 Using the MONSUM.COM Procedure

The procedure in Example 17–4 produces daily and prime-time clusterwide summaries.

Example 17–4 MONSUM.COM Procedure

```
$ SET VERIFY
$ !
$ !  MONSUM.COM
$ !
$ !  This command file is to be placed in a cluster-accessible directory
$ !  called SYS$MONITOR and executed at the convenience of the cluster
$ !  manager.  The file generates both 24-hour and "prime time" cluster
$ !  summaries and resubmits itself to run each day at midnight.
$ !
$ SET DEF SYS$MONITOR
$ SET NOON
$ !
$ !  Compute file specification for MONSUM.COM and resubmit the file.
$ !
$ FILE = F$ENVIRONMENT("PROCEDURE")
$ FILE = F$PARSE(FILE,,,"DEVICE")+F$PARSE(FILE,,,"DIRECTORY")+F$PARSE(FILE,,,"NAME")
$ SUBMIT 'FILE' /AFTER=TOMORROW /NOPRINT
$ !
$ !  Generate 24-hour cluster summary.
$ !
$ !
$ MONITOR/INPUT=(SYS$MONITOR:*MON*.DAT;*,MON$ARCHIVE:*MON*.DAT;*) -
  /NODISPLAY/SUMMARY=MONSUM.SUM -
  ALL_CLASSES+DISK/ITEM=ALL+SCS/ITEM=ALL-
  /BEGIN="YESTERDAY+0:0:0.00" /END="TODAY+0:0:0.00" /BY_NODE
```

(continued on next page)

Example 17–4 (Cont.) MONSUM.COM Procedure

```
$ !
$ !
$ !  Mail 24-hour summary file to cluster manager and delete the file from
$ !  SYS$MONITOR.
$ !
$ !
$ MAIL/SUBJECT="Daily Monitor Clusterwide Summary" MONSUM.SUM CLUSTER_MANAGER
$ DELETE MONSUM.SUM;*
$ !
$ !  Generate prime-time cluster summary.
$ !
$ !
$ MONITOR/INPUT=(SYS$MONITOR:*MON*.DAT;*,MON$ARCHIVE:*MON*.DAT;*) -
  /NODISPLAY/SUMMARY=MONSUM.SUM -
  ALL_CLASSES+DISK/ITEM=ALL+SCS/ITEM=ALL-
  /BEGIN="YESTERDAY+9:0:0.00" /END="YESTERDAY+18:0:0.00" /BY_NODE
$ !
$ !
$ !  Mail prime-time summary file to cluster manager and delete the file
$ !  from SYS$MONITOR.
$ !
$ !
$ MAIL/SUBJECT="Prime-Time Monitor Clusterwide Summary" MONSUM.SUM CLUSTER_MANAGER
$ DELETE MONSUM.SUM;*
$ !
$ !  End of MONSUM.COM
$ !
```

Note that MAIL commands in this procedure send files to user CLUSTER_MANAGER. Replace CLUSTER_MANAGER with the appropriate user name or logical name for your site.

Because summary data might be extensive, Digital recommends that you print out summary files.

18
Tracking Resource Use

This chapter describes how to find out how your system resources have been used. You can use this information to:

- Charge users for the resources they have used. You can produce reports of the resources used by individual users.
- Plan your future equipment requirements. You can monitor changing patterns of resource use and predict future demands.
- Troubleshoot the system. You can check the final exit status of processes.
- Improve system performance. You can find out the load that individual images and processes place on your system.
- Detect security breaches. You can identify unusual patterns of resource use.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Determining which resources are being tracked	Section 18.2
Controlling which resources are tracked	Section 18.3
Starting up a new accounting file	Section 18.4
Moving the accounting file	Section 18.5
Producing reports of resource use	Section 18.6
Setting up accounting groups	Section 18.7
Monitoring disk space	Section 18.8

This chapter explains the following concept:

Concept	Section
Accounting files	Section 18.1

18.1 Understanding Accounting Files

The system gathers information on resource use. For example, the information can include the resources such as CPU time used by each print job. The system stores this information in **accounting files**.

The resources tracked by default depend on the model of computer you use. However, you can control which resources are tracked. If you do not want to track resource use, you can stop the accounting file tracking resource use altogether (see Section 18.3).

Each node in a VMScluster has its own accounting file, known as its **current accounting file**. By default, this file is SYS$MANAGER:ACCOUNTNG.DAT, but you can control which file is used (see Section 18.5).

The information in the accounting files is in binary. You cannot display it with the TYPE command. To display the information, you use the Accounting utility (see Section 18.6).

18.2 Determining Which Resources Are Being Tracked

To determine which resources are currently being tracked, use the SHOW ACCOUNTING command:

```
$ SHOW ACCOUNTING
```

This command produces a screen display (see the example) that contains keywords in the following two categories:

- Keywords that show which types of resource are being tracked:

Keyword	Type of Resource
IMAGE	Resources used by an image
LOGIN_FAILURE	Resources used by an unsuccessful attempt to log in
MESSAGE	Unformatted resource record written to the accounting file by a call to the $SNDJBC system service
PRINT	Resources used by a print job
PROCESS	Resources used by a process

- Keywords that show which types of process are being tracked. When the resources for processes or images are tracked, these keywords show the process type:

Keyword	Type of Process
BATCH	Batch process
DETACHED	Detached process
INTERACTIVE	Interactive process
NETWORK	Network process
SUBPROCESS	Subprocess (the parent process can be a batch, detached, interactive, or network process)

Example

```
$ SHOW ACCOUNTING

Accounting is currently enabled to log the following activities:

      PROCESS        any process termination
      IMAGE          image execution
      INTERACTIVE    interactive job termination
      LOGIN_FAILURE  login failures
      NETWORK        network job termination
      PRINT          all print jobs
```

The keywords in this example show that the local node is tracking the resources used by each:

- Interactive and network process
- Image running in an interactive or network process
- Login failure
- Print job

18.3 Controlling Which Resources Are Tracked

You can control which resources the system tracks. To save disk space, you can stop the system tracking resources you are not interested in.

How to Perform This Task

1. Use the SET ACCOUNTING command with the /ENABLE and /DISABLE qualifiers in the following format to control which resources are tracked:

   SET ACCOUNTING/DISABLE[=(keyword[,...])]/ENABLE[=(keyword[,...])]

   The keywords are the same as those explained in Section 18.2.

2. If you want to make this change permanent, edit the SET ACCOUNTING command in the SYS$MANAGER:SYSTART_VMS.COM startup file.

Example

This example prevents the tracking of all resources except those used by interactive and batch processes:

```
$ SET ACCOUNTING/DISABLE/ENABLE=(PROCESS,INTERACTIVE,BATCH)
```

The /DISABLE qualifier is not followed by any keywords. Therefore, it disables the tracking of all resources. The /ENABLE qualifier then enables the tracking of the resources used by interactive and batch processes.

18.4 Starting Up a New Accounting File

To start up a new current accounting file, use the following command:

```
$ SET ACCOUNTING/NEW_FILE
```

This closes the current accounting file and opens a new version of it.

If the system encounters an error when trying to write information to the current accounting file, it automatically closes the file and opens a new version of it.

Example

This example closes the current accounting file, opens a new version of it, and changes the name of the old file to WEEK_24_RESOURCES.DAT. You can retain this file as a record of the resources used in that week.

```
$ SET ACCOUNTING/NEW_FILE
$ RENAME SYS$MANAGER:ACCOUNTNG.DAT;-1 WEEK_24_RESOURCES.DAT
```

18.5 Moving the Accounting File

When you first install your system, the current accounting file is SYS$MANAGER:ACCOUNTNG.DAT.

This file can become quite large. Moving it from your system disk can improve system performance.

How to Perform This Task

1. Define the logical name ACCOUNTNG in your system logical name table to point to the file you want to use. For example:

   ```
   $ DEFINE ACCOUNTNG MYDISK:MYFILE.DAT/SYSTEM
   ```

   Give the full file specification, including the device and directory.

   Note

   Two nodes cannot log information in the same accounting file. If you define ACCOUNTNG on two nodes to point to the same file, each node will open and use its own version of the file.

2. To make the change permanent, add this definition to the file SYS$MANAGER:SYLOGICALS.COM.
3. Use the SET ACCOUNTING command with the /NEW_FILE qualifier to create and use the new file:

   ```
   $ SET ACCOUNTING/NEW_FILE
   ```

Example

This example changes the current accounting file to MYDISK:MYFILE.DAT.

```
$ DEFINE ACCOUNTNG MYDISK:MYFILE.DAT/SYSTEM
$ SET ACCOUNTING/NEW_FILE
```

18.6 Producing Reports of Resource Use

There are three types of report:

Type of Report	Qualifier
Brief	/BRIEF (the default)
Full	/FULL
Summary	/SUMMARY

To produce a report, use the ACCOUNTING command with the appropriate qualifier in the following format:

ACCOUNTING [filespec[,...]/qualifier[,...]]

This runs the Accounting utility. The *filespec* parameter lists the accounting files you want to process. If you omit it, the Accounting utility processes the default current accounting file, SYS$MANAGER:ACCOUNTNG.DAT.

By default, the Accounting utility processes all the records in the accounting files you specify. You can use selection qualifiers to specify which records you want to process.

By default, brief and full reports present the records in the order in which they were logged in the accounting file. When you produce brief and full reports, you can use the /SORT qualifier to specify another order.

Example

This example produces a brief report of the information in the file that the logical name ACCOUNTNG points to. The /TYPE qualifier selects records for print jobs only. The /SORT qualifier displays them in reverse alphabetical order of user name.

```
$ ACCOUNTING ACCOUNTNG/TYPE=PRINT/SORT=-USER

    Date / Time      Type  Subtype  Username      ID      Source Status
-------------------------------------------------------------------------
13-SEP-1994 13:36:04 PRINT          SYSTEM     20A00442          00000001
13-SEP-1994 12:42:37 PRINT          JONES      20A00443          00000001
13-SEP-1994 14:43:56 PRINT          FISH       20A00456          00000001
14-SEP-1994 19:39:01 PRINT          FISH       20A00265          00000001
14-SEP-1994 20:09:03 PRINT          EDWARDS    20A00127          00000001
14-SEP-1994 20:34:45 PRINT          DARNELL    20A00121          00000001
14-SEP-1994 11:23:34 PRINT          CLARK      20A0032E          00040001
14-SEP-1994 16:43:16 PRINT          BIRD       20A00070          00040001
14-SEP-1994 09:30:21 PRINT          ANDERS     20A00530          00040001
```

18.7 Setting Up Accounting Groups

Users are already organized into UIC security groups. For accounting purposes, security groups are often inappropriate. You can put users into accounting groups with the Authorize utility using the /ACCOUNT qualifier. In this way, each user is in an accounting group and a security group.

Using the Accounting utility, you can:

- Summarize the resources used by all the users in a particular accounting or security group. To do this, use the ACCOUNT or UIC keyword with the /SUMMARY qualifier.
- Select records for all the users in a particular accounting or security group. To do this, use the /ACCOUNT or /UIC qualifier.

How to Perform This Task

1. Plan your accounting groups. Decide which users you want in each accounting group, and choose names for the groups.

   The name of an accounting group can be a maximum of eight characters long.

2. Change the account field values in the UAF. Use the Authorize utility's MODIFY command in the following format to change the value in the account field to the name of the user's accounting group:

   MODIFY username/ACCOUNT=accounting-group-name

   where:

   - *username* is the name of the user
   - *accounting-group-name* is the name of the accounting group that you want that user to be in

The next time your users log in, they will be in their new accounting groups, and their resource use will be tagged with the appropriate accounting group names.

Example

This example modifies the accounting group name to SALES_W8 for the username FORD:

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF> MODIFY FORD/ACCOUNT=SALES_W8
UAF> EXIT
```

18.8 Monitoring Disk Space

To find out how much disk space a user is using, use SYSMAN or, if you have not enabled disk quotas, the DIRECTORY command.

How to Perform This Task

Use either of the following methods:

- Use the SYSMAN command DISKQUOTA SHOW in the following format:

  DISKQUOTA SHOW uic [/DEVICE=diskname]

  This shows the number of blocks used by all the files that are owned by the specified user on the specified disk.

- Use the DIRECTORY command with the /SIZE and /GRAND_TOTAL qualifiers in the following format:

  DIRECTORY diskname:[username...]/SIZE=ALLOCATION/GRAND_TOTAL

  This shows the number of blocks used by all the files in and under the specified user's root directory.

  Note that the DIRECTORY command does not include the blocks used by file headers or the user's root directory.

Examples

1. This example uses SYSMAN to find out the number of blocks used by all the files that are owned by each user.

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA SHOW *

%SYSMAN-I-QUOTA, disk quota statistics on device SYS$SYSTEM:MYDISK
Node UNION
     UIC                  Usage        Permanent Quota   Overdraft Limit
[0,0]                     0            1000              100
[DOC,EDWARDS]             115354       150000            5000
[DOC,FISH]                177988       250000            5000
[DOC,SMITH]               140051       175000            5000
[DOC,JONES]               263056       300000            5000
```

2. This example uses the DIRECTORY command to show the number of blocks used by all the files in and under MYDISK:[PARSONS].

```
$ DIRECTORY MYDISK:[PARSONS...]/SIZE=ALLOCATION/GRAND_TOTAL

Grand total of 28 directories, 2546 files, 113565 blocks.
```

19
VMScluster Considerations

This chapter describes concepts related to the VMScluster environment; it also tells how to use the Show Cluster utility (SHOW CLUSTER) to display information about your cluster and the System Management utility (SYSMAN) to manage your VMScluster environment.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Setting up a VMScluster environment	Section 19.1.1
Beginning to use Show Cluster utility commands	Section 19.2.2
Adding information to a Show Cluster report	Section 19.2.3
Controlling the display of Show Cluster data	Section 19.2.4
Formatting the display of Show Cluster data	Section 19.2.5
Creating a Show Cluster utility startup initialization file	Section 19.2.6
Using command procedures containing Show Cluster utility commands	Section 19.2.7
Using the System Management utility to manage security and system time	Section 19.4
Using System Management utility DCL commands to manage your VMSclusters	Section 19.5

This chapter explains the following concepts:

Concept	Section
VMScluster systems	Section 19.1
Clusterwide system management	Section 19.1.2
The Show Cluster utility	Section 19.2.1
The System Management utility and VMScluster management	Section 19.3

19.1 Understanding VMScluster Systems

A **VMScluster** system is a loosely coupled configuration of two or more computers and storage subsystems. A VMScluster system appears as a single system to the user even though it shares some or all of the system resources. When a group of computers shares resources clusterwide, the storage and computing resources of all of the computers are combined, which can increase the processing capability, communications, and availability of your computing system.

A **shared resource** is a resource (such as a disk) that can be accessed and used by any node in a VMScluster system. Data files, application programs, and printers are just a few items that can be accessed by users on a cluster with shared resources, without regard to the particular node on which the files or program or printer might physically reside.

When disks are set up as shared resources in a VMScluster environment, users have the same environment (password, privileges, access to default login disks, and so on) regardless of the node that is used for logging in. You can realize a more efficient use of mass storage with shared disks, because the information on any device can be used by more than one node—the information does not have to be rewritten in many places. When you use the OpenVMS **mass storage control protocol** (MSCP) server software, disks and tapes can be made accessible to nodes that are not directly connected to the storage devices.

VAX

On VAX systems, you can also use the **tape mass storage control protocol** (TMSCP) server software to make disks and tapes accessible to nodes that are not directly connected to the storage devices. ♦

On AXP and VAX systems, you can also set up print and batch queues as shared resources. In a VMScluster configuration with shared print and batch queues, a single queue database manages the queues for all nodes. The queue database makes the queues available from any node. For example, suppose your VMScluster configuration has fully shared resources and includes nodes ALBANY, BASEL, and CAIRO. A user logged in to node ALBANY can send a file that physically resides on node BASEL to a printer that is physically connected to node CAIRO, and the user never has to specify (or even know) the nodes for either the file or the printer.

A number of types of VMScluster configurations are possible. Refer to *VMScluster Systems for OpenVMS* and either the VMScluster or VAXcluster Software Product Description (SPD) for complete information about supported devices and configurations.

The following sections briefly describe VMScluster systems. For complete information about setting up and using a VMScluster environment, see *VMScluster Systems for OpenVMS*.

19.1.1 Setting Up a VMScluster Environment

Once you have planned your configuration, installed the necessary hardware, and checked hardware devices for proper operation, you can set up a VMScluster system using various system software facilities. Setup procedures to build your VMScluster system follow.

Procedure	For More Information
Installing or upgrading the operating system on the first VMScluster computer	Installation and operations guide for your computer
Installing required software licenses	*OpenVMS License Management Utility Manual*
Configuring and starting the DECnet for OpenVMS network	*DECnet for OpenVMS Networking Manual*
Preparing files that define the cluster operating environment and that control disk and queue operations	*VMScluster Systems for OpenVMS*

Procedure	For More Information
Adding computers to the cluster	*VMScluster Systems for OpenVMS*

Depending on various factors, the order in which these operations are performed can vary from site to site, as well as from cluster to cluster at the same site.

19.1.2 Clusterwide System Management

Once any system is installed, the system manager must decide how to manage users and resources for maximum productivity and efficiency while maintaining the necessary security. VMScluster systems provide the flexibility to distribute users and resources to suit the needs of the environment. VMScluster system resources can also be easily redistributed as needs change. Even with the vast number of resources available, the VMScluster configuration can be managed as a single system.

VMScluster system managers have several tools and products to help them manage their systems as a unified entity.

VMScluster Tools

The following utilities are provided with the operating system:

Utility	Description
System Management utility (SYSMAN)	Allows the system manager to send common control commands across all, or a subset of, the nodes in the VMScluster system. (Described in Section 19.5.)
Show Cluster utility (SHOW CLUSTER)	Monitors activity in a VMScluster configuration, and then collects and sends information about that activity to a terminal or other output device. (Described in Section 19.2.)

VAX

System Management Applications (VAX Only)

On VAX systems, the following products are *not* provided with the operating system:

Product	Description
VAXcluster Console System (VCS)	Designed to consolidate the console management of the VAXcluster system at a single console terminal.
VAX Performance Advisor (VPA)	Assists the VAXcluster system manager in the performance analysis and capacity planning of the VAXcluster system.
VAX Storage Library System (SLS)	A set of software tools that enables a system manager to manage collections of removable media, including magnetic tape, cartridge tape, and optical disks.
DECscheduler for VMS	A distributed, automatic scheduling application for the systems in a VAXcluster configuration.
DECperformance Solution (DECps)	A set of software tools that provide basic user accounting and chargeback reports, a capacity planning tool, and a performance management tool that identifies and recommends solutions to bottlenecks.

Additional information about these system management tools can be found in the appropriate product documentation. ♦

19.2 Using the Show Cluster Utility (SHOW CLUSTER)

The Show Cluster utility (SHOW CLUSTER) monitors nodes in a VMScluster system. You can use the utility to display information about cluster activity and performance.

The following sections describe the Show Cluster utility and explain how to perform these operations:

Operation	For More Information
Begin to use SHOW CLUSTER commands	Section 19.2.2
Add information to a SHOW CLUSTER report	Section 19.2.3
Control the display of data	Section 19.2.4
Create a SHOW CLUSTER startup initialization file	Section 19.2.6
Use command procedure containing SHOW CLUSTER commands	Section 19.2.7

19.2.1 Understanding SHOW CLUSTER

You can display SHOW CLUSTER information on your terminal screen or send it to a device or a file. You can use the Show Cluster utility interactively, with command procedures, or with an initialization file in which you define default settings. Because this utility is installed with the CMKRNL privilege, SHOW CLUSTER requires no special privilege.

SHOW CLUSTER information includes approximately 100 fields of data. You can customize the appearance of SHOW CLUSTER reports or define reports for access to often-needed data.

SHOW CLUSTER reports are organized by classes and fields:

Unit of Organization	Description
Class	Group of related fields of information. You can use class names to selectively add or remove an entire class from a report. Each class displays certain fields by default. Some classes have additional fields that you can add or remove using the field name.
Field	Column of data in a report. You use a unique field name to refer to each field of data. You can use the field name to selectively add or remove a field from reports. For the names and descriptions of all of the fields in each class, see the ADD (Field) command in the *OpenVMS System Management Utilities Reference Manual*.

You can add fields or classes to the default SHOW CLUSTER report. If you add a field or class to a report in a continuous display, SHOW CLUSTER automatically adds the new data to the display.

Example 19–1 shows a sample default SHOW CLUSTER report. The default report has two classes of information: SYSTEMS and MEMBERS. Below each class name are columns of fields that are associated with each class of information.

Example 19–1 SHOW CLUSTER Default Display

```
View of Cluster from system ID 77777  node: CLUB
+-----------------------------------------+---------+
|                 SYSTEMS                 | MEMBERS |
+--------+----------------------+----------+---------+
|  NODE  |       HW_TYPE        | SOFTWARE |  STATUS |
+--------+----------------------+----------+---------+
| CLUB   | DEC 4000 Model 610   | VMS X5EM | MEMBER  |
| DISK12 | RF72                 | RFX T251 |         |
| CONS07 | EVAX                 | CON V1.0 |         |
| DISK14 | RF72                 | RFX V255 |         |
| CHIP   | DEC 4000 Model 620   | VMS X5EM | MEMBER  |
| DISK3  | RF72                 | RFX V254 |         |
| DISK1  | RF72                 | RFX V256 |         |
| SPREE  | DEC 3000 Model 500   | VMS X5EM | MEMBER  |
| SPRITZ | VAX 4000-300         | VMS A5.5 | MEMBER  |
+--------+----------------------+----------+---------+
```

Table 19–1 briefly describes the fields shown in Example 19–1.

Table 19–1 Fields in Default SHOW CLUSTER Report

Field Name	Description
NODE	Node name of the remote system. Normally, the cluster manager sets the node name using the SYSGEN parameter SCSNODE. The node name should be the same as the DECnet for OpenVMS node name.
HW_TYPE	Hardware type and model of the remote system.
SOFTWARE	Name and version of the operating system currently running on the remote system.
STATUS	Status of the node in the cluster. (MEMBER indicates that the system is participating in the cluster.)

Over time, you can determine the most valuable classes and fields of data for your SHOW CLUSTER reports; you can then create a startup initialization file that establishes your default report formats. You can also build command procedures to use while running SHOW CLUSTER interactively. In this way, you can quickly reformat the report to show the data that is relevant for your installation. Startup initialization files and command procedures are explained later in this chapter.

Because SHOW CLUSTER information includes many fields of data, the report can quickly extend beyond screen limits. Therefore, SHOW CLUSTER provides mechanisms to help you control the display of data, including the following:

- 38 SHOW CLUSTER commands
- A default keypad, which you can redefine

These mechanisms are described in detail in the *OpenVMS System Management Utilities Reference Manual.*

19.2.2 Beginning to Use SHOW CLUSTER Commands

To use the Show Cluster utility, you enter the SHOW CLUSTER command. If you specify the command without any qualifiers, however, SHOW CLUSTER simply displays a default report like that shown in Example 19–1 and then displays the DCL prompt.

In a continuous display, on the other hand, you can enter SHOW CLUSTER commands to control report output. You can, for example, add classes or fields to, or remove classes or fields from, reports. To invoke a continuous display, in which you can enter SHOW CLUSTER commands, you need to use the /CONTINUOUS qualifier on the SHOW CLUSTER command. (SHOW CLUSTER command qualifiers are described in Section 19.2.2.3.)

How to Perform This Task

To invoke a continuous display of default SHOW CLUSTER report information, enter the following command:

```
$ SHOW CLUSTER/CONTINUOUS
```

SHOW CLUSTER then displays a default report. By default, SHOW CLUSTER updates the display every 15 seconds, with the changed data displayed in reverse video. After the default report, SHOW CLUSTER displays the following prompt:

```
Command>
```

(If the report extends below the limit of your terminal screen and you do not see the Command> prompt, you can press Return to display the prompt.)

The following sections contain instructions for performing beginning SHOW CLUSTER tasks:

Operation	For More Information
View information that is off the screen	Section 19.2.2.1
Exit from a continuous display	Section 19.2.2.2
Use SHOW CLUSTER qualifiers	Section 19.2.2.3

19.2.2.1 Viewing Information That Is Off the Screen

The PAN command allows you to view the entire display by shifting your view of the display by column (horizontally) or by line (vertically).

Note

Report headings also move out of view as the reports in the display are panned beyond the limits of the screen. The SCROLL command, which is explained in Section 19.2.5.4, preserves the headings as you scroll information. To use the SCROLL command, you must take the additional step of selecting a report if you have more than one report on the screen.

How to Perform This Task

To pan the display, do one of the following:

- Enter PAN commands at the command prompt; for example:

  ```
  Command> PAN DOWN 10
  ```

  The command in this example moves the display down 10 lines.

- Define the arrow keys as PAN commands:

```
Command> SET FUNCTION PAN
```

This command redefines the arrow keys as follows:

Arrow Key	Redefinition
↑	PAN UP 1
↓	PAN DOWN 1
→	PAN RIGHT 1
←	PAN LEFT 1

You can then use the arrow keys to move up, down, right, and left in the display.

See the SET FUNCTION and PAN commands in the *OpenVMS System Management Utilities Reference Manual* for details.

Resetting Arrow Keys

By default, the SHOW CLUSTER arrow keys are set to the EDIT function. This means that, at the command prompt, you can perform command line editing that is similar to DCL line-mode editing. For example, the left arrow key moves the cursor to the left, and the up arrow key recalls the previous command. See the *OpenVMS User's Manual* for information on DCL line-mode editing.

When you use the SET FUNCTION command, you reset the function keys. After that, the arrow keys are redefined and DCL line-mode editing is disabled.

To reset the arrow keys, enter the following command:

```
Command> SET FUNCTION EDIT
```

19.2.2.2 Exiting from a Continuous Display

To exit from a continuous display, do one of the following:

- To return to the DCL prompt, do one of the following:
  - Enter EXIT after the Command> prompt.
  - Press Ctrl/Z.
  - Press Ctrl/Y.
- To exit without erasing the screen, press Ctrl/C.

19.2.2.3 Using SHOW CLUSTER Qualifiers

Table 19–2 briefly describes the qualifiers you can use with the SHOW CLUSTER command. The *OpenVMS System Management Utilities Reference Manual* contains reference information about these SHOW CLUSTER qualifiers.

Table 19–2 SHOW CLUSTER Qualifiers

Qualifier	Function
/BEGINNING=*time*	Specifies the time that the SHOW CLUSTER session is to begin.
/CONTINUOUS	Controls whether SHOW CLUSTER runs as a continuously updating display.
/ENDING=*time*	Specifies the time that the SHOW CLUSTER session is to end.
/INTERVAL=*seconds*	Specifies the number of seconds that report information remains on the screen before it is updated.
/OUTPUT=*file-spec*	Directs the output from SHOW CLUSTER to the specified file instead of to the current SYS$OUTPUT device.

Example

In a continuous display, SHOW CLUSTER updates the display every 15 seconds by default. You can change this interval by using the /INTERVAL qualifier.

```
$ SHOW CLUSTER/CONTINUOUS/INTERVAL=5
```

In this example, SHOW CLUSTER updates reports every 5 seconds, displaying changed data in reverse video.

19.2.3 Adding Information to a Report

When you use the SHOW CLUSTER command, the resulting report is only part of the total information available. As shown in Example 19–1, the default classes displayed are MEMBERS and SYSTEMS. Table 19–3 briefly describes all the classes you can display in SHOW CLUSTER reports. See the *OpenVMS System Management Utilities Reference Manual* for details about these classes.

Table 19–3 Classes of Information Available in SHOW CLUSTER Reports

Classes	Information Displayed
CIRCUITS	Describes virtual circuits on VMScluster systems.
CLUSTER	Shows general information about the VMScluster system, such as the time it was formed, the last time a system joined or left, and the VMScluster quorum.
CONNECTIONS	Describes the connections established over a virtual circuit in the VMScluster system
COUNTERS	Shows counts of the total accumulated traffic over a connection for the life of the connection.
CREDITS	Shows send and receive credit counts for connections in the VMScluster system.
ERRORS	Displays a count of the errors on each port, along with information on the feasibility of reinitializing a port.
LOCAL_PORTS	Displays information on the local system interface to the VMScluster system, such as the name, number, and status of each port, and the number of entries in the queues associated with each port.
MEMBERS	Describes systems actively participating in the VMScluster system.

(continued on next page)

Table 19–3 (Cont.) Classes of Information Available in SHOW CLUSTER Reports

Classes	Information Displayed
SYSTEMS	Describes all VMScluster systems. It shows node name, identification number, hardware type, and software version.

Example

The following example shows how to add the CLUSTER class to a SHOW CLUSTER display:

```
Command> ADD CLUSTER
```

Example 19–2 shows the display that results from entering the ADD CLUSTER command.

Example 19–2 SHOW CLUSTER Display with CLUSTER Report

```
View of Cluster from system ID 77777  node: CLUB
+-------------------------------------------+---------+
|                 SYSTEMS                   | MEMBERS |
+--------+---------------------+----------+---------+
|  NODE  |       HW_TYPE       | SOFTWARE |  STATUS |
+--------+---------------------+----------+---------+
| CLUB   | DEC 4000 Model 610  | VMS X5EM | MEMBER  |
| DISK12 | RF72                | RFX T251 |         |
| CONS07 | EVAX                | CON V1.0 |         |
| DISK14 | RF72                | RFX V255 |         |
| CHIP   | DEC 4000 Model 620  | VMS X5EM | MEMBER  |
| DISK3  | RF72                | RFX V254 |         |
| DISK1  | RF72                | RFX V256 |         |
| SPREE  | DEC 3000 Model 500  | VMS X5EM | MEMBER  |
| SPRITZ | VAX 4000-300        | VMS A5.5 | MEMBER  |
+--------+---------------------+----------+---------+

+--------------------------------------+
|               CLUSTER                |
+-----------+----------+---------------+
| CL_QUORUM | CL_VOTES |    QD_NAME    |
+-----------+----------+---------------+
|         2 |        3 |               |
+-----------+----------+---------------+
```

Table 19–1 describes the fields shown in the top section of the report shown in Example 19–2. Table 19–4 briefly describes the fields in the CLUSTER report.

Table 19–4 Fields in Sample CLUSTER Report

Field Name	Description
CL_QUORUM (Cluster quorum)	The number of votes that must be present for the cluster to function and permit user activity. CL_QUORUM is equal to (CL_EXPECTED_VOTES + 2) divided by 2.
CL_VOTES (Cluster votes)	Total number of votes contributed by all members of the cluster at any point in time.
QD_NAME (Quorum disk name)	Full device name of the quorum disk.

For detailed descriptions of the fields in the CLUSTER class, see the *OpenVMS System Management Utilities Reference Manual.*

19.2.4 Controlling the Display of Data

Using SHOW CLUSTER commands, you can remove fields or classes from a display, remove broadcast messages from the screen, and refresh the screen display at any time. The following sections explain how to perform these operations.

19.2.4.1 Entering Commands to Display Data

SHOW CLUSTER allows you to customize the display of data during a continuous session by entering various commands. The *OpenVMS System Management Utilities Reference Manual* describes SHOW CLUSTER commands in detail.

Updating of the continuous display stops as soon as you enter input from the terminal keyboard. When you press the Return key after entering a command, updating of the display resumes until you enter another command.

By default, updating takes place at 15-second intervals. If you do not enter a new command within 15 seconds, the command prompt disappears, and two more lines of data take its place.

19.2.4.2 Removing Broadcast Messages

When you receive a system broadcast message during a continuous SHOW CLUSTER session, the message appears at the bottom of the screen. A multiline message fills as many lines of the screen as it needs.

How to Perform This Task

The last broadcast message you receive remains on the screen until you acknowledge it by entering input from the terminal in one of the following ways:

- Press the Return key.
- Refresh the screen by pressing Ctrl/W.
- Enter a command.

If you receive more than one broadcast message, SHOW CLUSTER waits until the next update interval to display the next message.

SHOW CLUSTER also displays error messages at the bottom of the screen. For an explanation of the error messages, see the *OpenVMS System Messages and Recovery Procedures Reference Manual*.

19.2.4.3 Refreshing the Screen

Ordinarily, a continuous display is updated or refreshed according to the default or specified interval time. SHOW CLUSTER scans the software databases, extracts and stores data for each field, displays any new or changed data, and updates the time. On Digital and Digital-compatible terminals, reverse video highlights any changed data.

How to Perform This Task

You can refresh the screen at any time by one of the following methods:

- Modify the format of the display with the ADD, REMOVE, INITIALIZE, or SET command.
- Use the REFRESH command.
- Press Ctrl/W.

19.2.5 Formatting the Display of Data

Because SHOW CLUSTER allows you to include additional fields and classes, you can produce reports that overflow the physical limits of the terminal screen. However, you can use a number methods to modify the display to meet your needs:

Formatting Method	For More Information
Remove data from reports	Section 19.2.5.1
Modify field and screen size	Section 19.2.5.2
Move a report	Section 19.2.5.3
Scroll a report	Section 19.2.5.4

19.2.5.1 Removing Information from a Report

You might want to remove certain fields or classes to reduce the width of a report to fit the limits of your screen. Also, certain fields or classes might not be important for your particular needs. You can also remove particular types of data to reduce the length of the report.

How to Perform This Task

You use the REMOVE command to remove entire fields and classes, or subsets of fields and classes. To remove subsets of data, use the appropriate qualifier with the REMOVE *class-name* command. See the REMOVE commands in the *OpenVMS System Management Utilities Reference Manual* for appropriate class names and qualifiers.

Examples

1. `Command> REMOVE SOFTWARE`

   The command in this example removes the SOFTWARE field from the SHOW CLUSTER report shown in Example 19–1.

   See the ADD (Field) command description in the *OpenVMS System Management Utilities Reference Manual* for a list of valid field names.

2. `Command> REMOVE MEMBERS`

   The command in this example removes the MEMBERS class from the SHOW CLUSTER report shown in Example 19–1.

19.2.5.2 Modifying Field and Screen Size

To make a report fit the physical limits of the screen, you can change the width of certain fields in the report. For example, if SHOW CLUSTER provides a field width that can contain any possible value and the values your cluster generates do not require that much space, you can adjust the field width with the SET (Field) command.

SHOW CLUSTER also allows you to adjust the size of the terminal screen. If the terminal is Digital-compatible and supports a wide report, you can set the screen to a width of up to 511 columns by specifying an appropriate value to the SET SCREEN command.

Examples

1. `Command> SET TRANSITION_TYPE/WIDTH=10`

   The command in this example sets the width of the TRANSITION_TYPE field to 10, which removes the time of day from the field but leaves the date.

2. `Command> SET SCREEN=132`

   The command in this example sets the screen width to 132.

Refer to the *OpenVMS System Management Utilities Reference Manual* for more details about using the SET (Field) and SET SCREEN commands.

19.2.5.3 Moving a Report

By default, SHOW CLUSTER operates with AUTO_POSITIONING ON. This means that the utility automatically arranges the reports to take best advantage of the available display space. However, you can position reports manually with the MOVE command, which implicitly sets AUTO_POSITIONING to OFF.

If you have multiple reports in your display, you must first select the report to be repositioned. You use the command SELECT *window-name* to specify the report name; for example:

- SCS (the default report, which usually includes fields in the SYSTEMS and MEMBERS classes)
- CLUSTER
- LOCAL_PORTS

Note

To select any report except the default SCS report, you must first add the class to the display if it is not already displayed; for example:

```
Command> ADD LOCAL_PORTS
```

As an alternative, you can repeatedly press the Select function key or the period key on the keypad to cycle from one report to the next. The selected report appears highlighted.

How to Perform This Task

To move a report, do either of the following:

- Enter MOVE commands at the command prompt.
- Use the arrow keys that you define as MOVE commands.

  ```
  Command> SET FUNCTION MOVE
  ```

  This command redefines the arrow keys as follows:

Arrow Key	Redefinition
↑	MOVE UP 1
↓	MOVE DOWN 1
→	MOVE RIGHT 1
←	MOVE LEFT 1

When you enter a MOVE command, the display changes position by column (horizontally) or by line (vertically). For example, entering the command MOVE LEFT 5 moves the display 5 columns to the left. An empty frame appears around the new position of the report.

When you are satisfied with the position of the report, enter the DESELECT command, which moves the report to the new position. Entering another SELECT command before the previous MOVE operation has been deselected also moves the report to its new position.

Example

```
Command> SELECT CLUSTER
Command> MOVE RIGHT 10
Command> DESELECT
```

Following is an explanation of the commands in the example:

1. The SELECT command selects the CLUSTER report (which is then highlighted).
2. The MOVE command positions the report frame 10 spaces to the right.
3. The DESELECT command terminates the MOVE operation and displays the contents of the report.

For more information, see the SELECT, SET FUNCTION, and DESELECT commands in the *OpenVMS System Management Utilities Reference Manual*.

To reset the arrow keys, enter the following command:

```
Command> SET FUNCTION EDIT
```

19.2.5.4 Scrolling a Report

The SCROLL command provides a means of quickly scanning through a report without losing column headings. Scrolling scans a display by field (horizontally) and by line (vertically). The report headings remain stationary when you scroll vertically.

When the display has more than one report, you must first select a report by entering the SELECT command. The selected report is highlighted.

How to Perform This Task

To scroll a display, do either of the following:

- Enter SCROLL commands at the command prompt.
- Use the arrow keys that you define as SCROLL commands.

  ```
  Command> SET FUNCTION SCROLL
  ```

This command redefines the arrow keys as follows:

Arrow Key	Redefinition
↑	SCROLL UP 1
↓	SCROLL DOWN 1
→	SCROLL RIGHT 1
←	SCROLL LEFT 1

Example

```
Command> SELECT SCS
Command> SET FUNCTION SCROLL
```

The commands in this example first select the SCS report (which is then highlighted), and then set the arrow keys to scroll functions. See the SET FUNCTION and SCROLL commands in the *OpenVMS System Management Utilities Reference Manual* for more information.

To reset the arrow keys, enter the following command:

```
Command> SET FUNCTION EDIT
```

19.2.6 Creating a SHOW CLUSTER Startup Initialization File

To customize the SHOW CLUSTER display, you can create a startup initialization file, which the utility executes when you enter it. SHOW CLUSTER takes the original default display, and adds or removes whatever classes or fields you specify. The resulting display becomes your default startup format. A startup initialization file resembles the following:

```
!
!Startup Initialization File
!
!
INITIALIZE
REMOVE MEMBERS
ADD RP_REVISION,RP_TYPE,SYS_ID
SET SCREEN=132
```

This startup procedure causes SHOW CLUSTER to delete the MEMBERS class information from the default display. The procedure also adds the RP_REVISION and RP_TYPE fields from the CIRCUITS class and the SYS_ID field from the SYSTEMS class. The last line of the procedure sets the screen size to 132 columns.

How to Perform This Task

To create an initialization file, follow these steps:

1. Define the logical name SHOW_CLUSTER$INIT as *device:[directory]*SHCINI before invoking SHOW CLUSTER.

   For a startup file to execute before the display begins, you must assign the logical name SHOW_CLUSTER$INIT to the initialization file; for example:

   ```
   DEFINE SHOW_CLUSTER$INIT DEVA:[JONES]SHCINI
   ```

   When invoked, SHOW CLUSTER searches for the file defined by SHOW_CLUSTER$INIT. In this example, SHOW CLUSTER looks for DEVA:[JONES]SHCINI.INI when it starts up. If the initialization file is found, SHOW CLUSTER executes the procedure before beginning the display.

If you do not define SHOW_CLUSTER$INIT or it does not include a directory specification, SHOW CLUSTER searches the current default directory for a file named SHOW_CLUSTER.INI.

2. Customize the display using SHOW CLUSTER commands during a continuous SHOW CLUSTER session.
3. Preserve the command sequence by entering the following command:

```
Command> SAVE SHOW_CLUSTER$INIT.INI
```

   You must specify SHOW_CLUSTER$INIT.INI, because the SAVE command creates a file with a file type of .COM by default. SHOW CLUSTER looks for an .INI file when it searches for a startup initialization file.

You can edit the file that the SAVE command creates to include comments or to improve its efficiency. For more information, see the SAVE command in the *OpenVMS System Management Utilities Reference Manual.*

Instead of having SHOW CLUSTER build an initialization file, you can build one yourself in the same way you build a command procedure. The next section provides guidelines for creating a command procedure.

19.2.7 Using Command Procedures Containing SHOW CLUSTER Commands

You can create command procedures that contain SHOW CLUSTER commands. Such files let you modify display characteristics without having to enter commands interactively. You can use command procedures during a continuous SHOW CLUSTER session to perform a series of commands, for example, to customize the output of the display.

Following are guidelines for writing command procedures that contain SHOW CLUSTER commands:

- Use any valid SHOW CLUSTER commands.
- Nest command procedures up to 16 levels deep.
- Include the SHOW CLUSTER command INITIALIZE as the first command in the file. The INITIALIZE command ensures that the report is in a known state before any commands are executed to modify it.

Notes

Do not include an EXIT command at the end of the command procedure. The EXIT command terminates SHOW CLUSTER and erases the SHOW CLUSTER display before you can see it.

Also, do not run SHOW CLUSTER command procedures from a batch job.

The following command procedure customizes a report display:

```
!
! Include only the node field from the default display; show votes
! and quorum for each node and for the cluster as a whole.
!
INITIALIZE
REMOVE SOFTWARE,STATUS
ADD VOTES,QUORUM,CL_VOTES,CL_QUORUM
```

This command procedure removes the SOFTWARE and STATUS fields from the report and adds fields that provide information about the cluster quorum and votes.

To execute a command procedure during a continuous SHOW CLUSTER session, specify the Execute Procedure (@) command, along with the file name of the command procedure. The default file type for command procedure files is .COM.

Example

The following command executes a command procedure named SYSMOD.COM:

```
Command> @SYSMOD
```

In this example, the default file type .COM is assumed because the file type is omitted.

For more information on creating command procedures, see the SAVE command in the *OpenVMS System Management Utilities Reference Manual.*

19.3 Understanding SYSMAN and VMScluster Management

The System Management utility (SYSMAN) provides two kinds of support for VMScluster management:

- Cluster-specific commands, CONFIGURATION SET and CONFIGURATION SHOW, that you can use to manage security data and system time in a VMScluster
- Access to DCL-level commands with the DO command, which gives you the ability to apply a single DCL command across an entire VMScluster, rather than having to enter the command on each node

Each SYSMAN command requires a specific level of privilege. For more information on each command, see the *OpenVMS System Management Utilities Reference Manual.*

19.4 Using SYSMAN to Manage Security and System Time

You can manage security data and system time for a VMScluster with the SYSMAN CONFIGURATION commands. Table 19–5 summarizes these CONFIGURATION commands and their functions.

Table 19–5 SYSMAN CONFIGURATION Commands

Command	Function
CONFIGURATION SET CLUSTER_AUTHORIZATION	Modifies the group number and password in a local area VMScluster
CONFIGURATION SHOW CLUSTER_AUTHORIZATION	Displays the group number and multicast address of a local area VMScluster
CONFIGURATION SET TIME	Updates system time
CONFIGURATION SHOW TIME	Displays current system time

19.4.1 Modifying the Group Number and Password

The group number identifies the group of nodes in the VMScluster, and the associated Ethernet address is used to send messages to all nodes in the cluster. The VMScluster password protects the integrity of the VMScluster membership.

Using the CONFIGURATION SET CLUSTER_AUTHORIZATION command modifies the group number and password, as recorded in SYS$SYSTEM:CLUSTER_AUTHORIZE.DAT. Normally, you do not need to alter records in the CLUSTER_AUTHORIZE.DAT file.

If your configuration has multiple system disks, SYSMAN automatically updates each copy of CLUSTER_AUTHORIZE.DAT, provided that you have defined the environment as a VMScluster with the SET ENVIRONMENT/CLUSTER command.

Caution

If you change either the group number or password, you must reboot the entire VMScluster.

You cannot display the VMScluster password for security reasons, but you can display the group number and group multicast address with the CONFIGURATION SHOW CLUSTER_AUTHORIZATION command.

Examples

1. The following command example sets the environment to a specific cluster, sets privilege to SYSPRV, and modifies the VMScluster password:

```
SYSMAN> SET ENVIRONMENT/CLUSTER/NODE=NODE21
SYSMAN> SET PROFILE/PRIVILEGE=SYSPRV
SYSMAN> CONFIGURATION SET CLUSTER_AUTHORIZATION/PASSWORD=GILLIAN
%SYSMAN-I-CAFOLDGROUP, existing group will not be changed
%SYSMAN-I-GRPNOCHG, Group number not changed
SYSMAN-I-CAFREBOOT, cluster authorization file updated.
The entire cluster should be rebooted.
```

2. The following command example displays the group number and multicast address for NODE21. Because the group number and password on other nodes in the VMScluster are identical, no further information is displayed.

```
SYSMAN> CONFIGURATION SHOW CLUSTER_AUTHORIZATION
Node NODE21: Cluster group number 65240
Multicast address: AB-00-04-01-F2-FF
```

19.4.2 Modifying the System Time

Use the CONFIGURATION SET TIME command to modify system time for nodes in a VMScluster, as well as for individual nodes. You can specify time values in the following format:

[dd-mmm-yyyy[:]] [hh:mm:ss.cc]

You can also enter delta time values. See the *OpenVMS User's Manual* for more information about time formats.

In a VMScluster environment, SYSMAN sets the time on each node to the value you specify. However, if you do not specify a value, SYSMAN reads the clock on the node from which you are executing SYSMAN and assigns this value to all nodes in the VMScluster. In a remote VMScluster, SYSMAN reads the clock on the target node in the cluster and assigns that value to all nodes. Note that the time-of-year clock is optional for some processors; see your processor's hardware handbook for more information.

SYSMAN tries to ensure that all processors in the VMScluster are set to the same time. Because of communication and processing delays, it is not possible to synchronize clocks exactly. However, the variation is typically less than a few hundredths of a second. If SYSMAN cannot set the time to within one-half second of the specified time, you receive a warning message that names the node that failed to respond quickly enough.

As a result of slight inaccuracies in each processor clock, times on various members of a VMScluster tend to drift apart. The first two examples show how to synchronize system time in a VMScluster.

Examples

1. The following procedure sets the time on all VMScluster nodes to the value obtained from the local time-of-year clock, waits 6 hours, then resets the time for the VMScluster:

```
$ SYNCH_CLOCKS:
$ RUN SYS$SYSTEM:SYSMAN
      SET ENVIRONMENT/CLUSTER
      CONFIGURATION SET TIME
      EXIT
$ WAIT 6:00:00
$ GOTO SYNCH_CLOCKS
```

2. The next example sets the environment to NODE21, NODE22, and NODE23, sets privilege, and modifies the system time on all three nodes:

```
SYSMAN> SET ENVIRONMENT/NODE=(NODE21,NODE22,NODE23)
SYSMAN> SET PROFILE/PRIVILEGE=LOG_IO
SYSMAN> CONFIGURATION SET TIME 12:38:00
```

3. The following example sets the environment to cluster and displays the system time for all nodes:

```
SYSMAN> SET ENVIRONMENT/CLUSTER/NODE=NODE23
SYSMAN> CONFIGURATION SHOW TIME
System time on node NODE21: 19-JUN-1994 13:32:19.45
System time on node NODE22: 19-JUN-1994 13:32:27.79
System time on node NODE23: 19-JUN-1994 13:32:58.66
```

19.5 Using the SYSMAN DO Command to Manage a VMScluster

Using the SYSMAN command DO enables you to execute a DCL command or command procedure on all nodes in the current environment. This is convenient when you are performing routine system management tasks on nodes in the VMScluster, such as:

- Installing images
- Starting up software
- Checking devices
- Checking memory

Each DO command executes as an independent process, so there is no process context retained between DO commands. For this reason, you must express all DCL commands in a single command string, and you cannot run a program that expects input.

In a VMScluster environment, SYSMAN executes the commands sequentially on all nodes in the VMScluster. Each command executes completely before SYSMAN sends it to the next node in the environment. Any node that is unable to execute the command returns an error message. SYSMAN displays an error message if the timeout period expires before the node responds.

In a dual-architecture heterogeneous VMScluster running both OpenVMS VAX and OpenVMS AXP, some uses of the DO command may require special handling. For example, if you are installing images that are named differently in each architecture, you can still use the DO command if you create logical name tables for VAX and for AXP nodes. See the example sequence that follows this description for an example.

Some DCL commands, such as MOUNT/CLUSTER or SET QUORUM/CLUSTER, operate clusterwide by design. It is best to avoid using these kinds of commands with the DO command in SYSMAN when the environment is set to cluster. As alternatives, you could leave SYSMAN temporarily with the SPAWN command and execute these commands in DCL, or you could define the environment to be a single node within the VMScluster.

Examples

1. The following example installs an image on a VMScluster. First, it adds CMKRNL and SYSPRV privileges to the current privileges because they are required by INSTALL and AUTHORIZE. The DO INSTALL command installs the file STATSHR. The DO MCR AUTHORIZE command sets up an account for user Jones, specifying a password and a default device and directory.

```
SYSMAN> SET PROFILE/PRIVILEGES=(CMKRNL,SYSPRV)/DEFAULT=SYS$SYSTEM
SYSMAN> DO INSTALL ADD/OPEN/SHARED WRKD$:[MAIN]STATSHR
SYSMAN> DO MCR AUTHORIZE ADD JONES/PASSWORD=COLUMBINE -
_SYSMAN> /DEVICE=WORK1/DIRECTORY=[JONES]
```

2. The following example sets the environment to cluster and starts up a software product called XYZ on each node in the VMScluster:

```
SYSMAN>SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, Current command environment:
        Clusterwide on local cluster
        Username SMITH    will be used on nonlocal nodes
SYSMAN> DO @SYS$STARTUP:XYZ_STARTUP
```

3. The following example shows how you can define logical names for VAX and AXP nodes in a dual-architecture heterogeneous VMScluster, so that you can use the DO command to install architecture-specific images.

```
$ CREATE/NAME_TABLE/PARENT=LNM$SYSTEM_DIRECTORY SYSMAN$NODE_TABLE
$ DEFINE/TABLE=SYSMAN$NODE_TABLE AXP_NODES NODE21,NODE22,NODE23
$ DEFINE/TABLE=SYSMAN$NODE_TABLE VAX_NODES NODE24,NODE25,NODE26
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/NODE=AXP_NODES
%SYSMAN-I-ENV, current command environment:
         Individual nodes: NODE21,NODE22,NODE23
         Username BOUCHARD will be used on nonlocal nodes
```

```
SYSMAN> DO INSTALL REPLACE SYS$LIBRARY:DCLTABLES.EXE
%SYSMAN-I-OUTPUT, command execution on node NODE21
%SYSMAN-I-OUTPUT, command execution on node NODE22
%SYSMAN-I-OUTPUT, command execution on node NODE23
SYSMAN> DO INSTALL REPLACE SYS$SYSTEM: DEC_FORTRAN.EXE
%SYSMAN-I-OUTPUT, command execution on node NODE21
%SYSMAN-I-OUTPUT, command execution on node NODE22
%SYSMAN-I-OUTPUT, command execution on node NODE23

SYSMAN> SET ENVIRONMENT/NODE=VAX_NODES
%SYSMAN-I-ENV, current command environment:
         Individual nodes: NODE24,NODE25,NODE26
         Username BOUCHARD will be used on nonlocal nodes

SYSMAN> DO INSTALL REPLACE SYS$LIBRARY:DCLTABLES.EXE
%SYSMAN-I-OUTPUT, command execution on node NODE24
%SYSMAN-I-OUTPUT, command execution on node NODE25
%SYSMAN-I-OUTPUT, command execution on node NODE26
SYSMAN> DO INSTALL REPLACE SYS$SYSTEM:FORTRAN$MAIN.EXE
%SYSMAN-I-OUTPUT, command execution on node NODE24
%SYSMAN-I-OUTPUT, command execution on node NODE25
%SYSMAN-I-OUTPUT, command execution on node NODE26
```

4. The following example shows which files are open on DISK2. You might use this if you want to dismount DISK2 and need to see which users in the VMScluster have files open.

```
SYSMAN >SET ENVIRONMENT/CLUSTER
%SYSMAN-I-ENV, Current command environment:
        Clusterwide on local cluster
        Username SMITH    will be used on nonlocal nodes
SYSMAN> DO SHOW DEVICE/FILES DISK2:

%SYSMAN-I-OUTPUT, command execution on node NODE21
Files accessed on device $1$DIA2: (DISK2, NODE22) on 14-JUL-1993 15:44:06.05
Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
%SYSMAN-I-OUTPUT, command execution on node NODE22
Files accessed on device $1$DIA2: (DISK2, NODE21) on 14-JUL-1993 15:44:26.93
Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
%SYSMAN-I-OUTPUT, command execution on node NODE23
Files accessed on device $1$DIA2: (NODE21, NODE22) on 14-JUL-1993 15:45:01.43
Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
%SYSMAN-I-OUTPUT, command execution on node NODE24
Files accessed on device $1$DIA2: (NODE22, NODE21) on 14-JUL-1993 15:44:31.30
Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
Susan Scott     21400059  [SCOTT]DECW$SM.LOG;228
_FTA7:          214000DD  [SCOTT]CARE_SDML.TPU$JOURNAL;1
%SYSMAN-I-OUTPUT, command execution on node NODE25
Files accessed on device $1$DIA2: (NODE21, NODE22) on 14-JUL-1993 15:44:35.50
Process name      PID     File name
                00000000  [000000]INDEXF.SYS;1
DECW$SESSION    226000E6  [SNOW]DECW$SM.LOG;6
_FTA17:         2260009C  [SNOW.MAIL]MAIL.MAI;1
SNOW_1          2260012F  [SNOW.MAIL]MAIL.MAI;1
SNOW_2          22600142  [SNOW.MAIL]MAIL.MAI;1
SNOW_3          22600143  [SNOW.MAIL]MAIL.MAI;1
```

5. The following example shows how much memory is available on the nodes in a VMScluster. You might use this if you are installing software and want to know if each node has enough memory available.

```
SYSMAN > SET ENVIRONMENT/NODE=(NODE21,NODE22)
%SYSMAN-I-ENV, Current command environment:
        Clusterwide on local cluster
        Username SMITH    will be used on nonlocal nodes
SYSMAN>  DO SHOW MEMORY
%SYSMAN-I-OUTPUT, command execution on node NODE21
              System Memory Resources on 14-JUL-1993 15:59:21.61
Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (64.00Mb)           131072       63955       65201        1916
Slot Usage (slots):                Total        Free    Resident     Swapped
  Process Entry Slots                360         296          64           0
  Balance Set Slots                  324         262          62           0
Fixed-Size Pool Areas (packets):   Total        Free      In Use        Size
  Small Packet (SRP) List          10568        1703        8865         128
  I/O Request Packet (IRP) List     3752         925        2827         176
  Large Packet (LRP) List            157          28         129        1856
Dynamic Memory Usage (bytes):      Total        Free      In Use     Largest
  Nonpaged Dynamic Memory        1300480       97120     1203360       60112
  Paged Dynamic Memory           1524736      510496     1014240      505408
Paging File Usage (pages):                      Free  Reservable       Total
  DISK$MTWAIN_SYS:[SYS0.SYSEXE]SWAPFILE.SYS
                                               10000       10000       10000
  DISK$MTWAIN_SYS:[SYS0.SYSEXE]PAGEFILE.SYS
                                               60502      -52278      100000
Of the physical pages in use, 19018 pages are permanently allocated to VMS.

%SYSMAN-I-OUTPUT, command execution on node NODE22
              System Memory Resources on 14-JUL-1993 15:59:42.65
Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (32.00Mb)            65536       44409       20461         666
Slot Usage (slots):                Total        Free    Resident     Swapped
  Process Entry Slots                240         216          24           0
  Balance Set Slots                  212         190          22           0
Fixed-Size Pool Areas (packets):   Total        Free      In Use        Size
  Small Packet (SRP) List           5080        2610        2470         128
  I/O Request Packet (IRP) List     3101        1263        1838         176
  Large Packet (LRP) List             87          60          27        1856
Dynamic Memory Usage (bytes):      Total        Free      In Use     Largest
  Nonpaged Dynamic Memory        1165312      156256     1009056      114432
  Paged Dynamic Memory           1068032      357424      710608      352368
Paging File Usage (pages):                      Free  Reservable       Total
  DISK$MTWAIN_SYS:[SYS1.SYSEXE]SWAPFILE.SYS
                                               10000       10000       10000
  DISK$MTWAIN_SYS:[SYS1.SYSEXE]PAGEFILE.SYS
                                              110591       68443      120000
Of the physical pages in use, 9056 pages are permanently allocated to VMS.
```

20
Network Considerations

As the manager of an OpenVMS system, you probably want to connect your system to a network by means of the DECnet interface. With this interface, you can link computers into flexible configurations to exchange information, share resources, and perform distributed processing.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Providing security for your node	Section 20.3
Providing host services	Section 20.4.1
Monitoring the network	Section 20.4.2
Testing the network	Section 20.4.3
Shutting down and restarting the network	Section 20.4.4

This chapter explains the following concepts:

Concept	Section
A DECnet for OpenVMS network	Section 20.1
How an OpenVMS system can be part of a network	Section 20.1.1
How nodes are connected to the network	Section 20.1.2
The configuration database	Section 20.1.3
How your system becomes a node in the network	Section 20.1.4
Preparations for joining a network	Section 20.2

For more details, refer to the following manuals:

Manual	Description
DECnet for OpenVMS Guide to Networking	Provides an introduction to networking on the system.
DECnet for OpenVMS Networking Manual	Includes conceptual and usage information.
DECnet for OpenVMS Network Management Utilities	Explains how to use the Network Control Program (NCP) utility.

Where appropriate, this chapter refers to specific manuals in this group.

20.1 Understanding DECnet for OpenVMS Networks

A **network** is a means of connecting computers that allows them to share or transfer information or communications. A network includes two or more computers that are connected, and the hardware and software that make those connections.

DECnet for OpenVMS is the name of the software and hardware products that, collectively, provide the means for various Digital operating systems to participate in a network. DECnet allows an OpenVMS operating system to function as a network node. As a part of a network, an OpenVMS system can communicate with all types of OpenVMS systems, as well as with many non-OpenVMS systems that support DECnet.

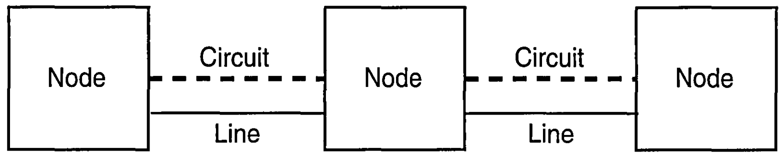
Table 20–1 defines terms related to DECnet networks.

Table 20–1 DECnet for OpenVMS Network Terminology

Term	Definition
Node	A computer system that is connected to another system in a network—by means of cables, telephone lines, microwave and satellite links, for example. An **adjacent node** is one that is connected to your node by a single physical line.
Line	Physical data path that connects adjacent nodes in a network. A **communications line** connects your computer to the DECnet network.
Circuit	Communications data path that connects adjacent nodes in a network. A circuit is not a physical data path but, rather, a logical connection that operates over a physical connection (a line). All input and output (I/O) between nodes takes place over circuits. You can configure a node to have a number of active circuits and lines that connect it to other systems in the network.
Logical link	Connects two processes and carries a stream of two-way communications traffic between the processes over a circuit. A single circuit established between two nodes can support many logical links concurrently.
Object	Process to which the logical link connects. Some objects are DECnet system programs—for example, the MAIL object; other objects are user-written programs. For two programs to communicate over the network, the source program on the local node establishes a logical link with the object on the remote node.
Ethernet	A single shared network channel, with all nodes having equal access to the channel. Ethernet offers local and remote connections as one integral network.

Figure 20–1 shows lines and circuits connecting nodes in a DECnet network.

Figure 20–1 Network Nodes, Circuits, and Lines



A DECnet network is decentralized. Many nodes connected to the network can communicate with each other without having to go through a central node. As a member of a multinode network, a node can communicate with any other network node, not merely with those nodes physically attached to it. This feature allows users to gain access to software facilities that might not exist on their particular nodes.

DECnet Routing

In a network of more than two nodes, the process of directing a data message from a source node to a destination node is called **routing**. DECnet supports **adaptive routing**, which routes messages through the network over the most cost-effective path. Adaptive routing also reroutes messages automatically if a circuit becomes disabled or a lower-cost path becomes available.

Nodes can be either routing nodes (called **routers**) or nonrouting nodes (known as **end nodes**). Both routers and end nodes can send messages to and receive messages from other nodes in the network. Following are the differences between a router and an end node.

- **Router**

  Routing node; has the ability to forward or route messages from itself to another node.

  A routing node can serve as an intermediate node on a path between two nodes exchanging messages, if the two nodes have no direct physical link to each other. Any node that has two or more active circuits connecting it to the network must be a router.

  DECnet supports routing within each area; DECnet also supports a second, higher level of routing that links the areas, resulting in less routing traffic throughout the network.

  The higher levels of routing are the following:

  – **Level 1 routers**

    These are nodes that perform routing within a single area.

  – **Level 2 routers** (or **area routers**)

    These are nodes that perform routing between areas as well as within their own area.

AXP

On AXP systems, DECnet does not support routing. This end-node only (nonrouting) capability means that an AXP node can receive packets addressed to it and can send packets to other nodes, but it cannot route packets. For more information on DECnet restrictions on AXP systems, see *A Comparison of System Management on OpenVMS AXP and OpenVMS VAX.*♦

- **End node**

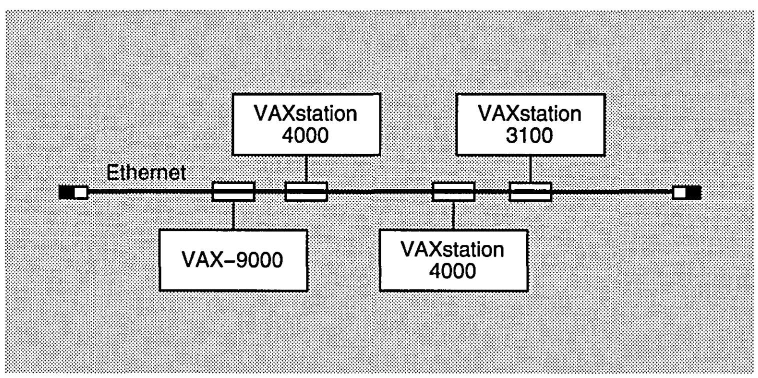
  Nonrouting node; can have only one active circuit connecting it to the network.

DECnet Configurations

DECnet supports configurations for both large and small networks. A typical small network might consist of two to four nodes. A maximum of 1023 nodes is possible in an undivided network, but the optimum number is approximately 200 to 300 nodes, depending on the topology.

Figure 20–2 illustrates a small Ethernet configuration of four nodes. Three VAXstation-based end nodes and one router (the VAX-9000) are connected to the Ethernet.

Figure 20–2 Example of a Small Local Area Network Configuration



ZK–6357–GE

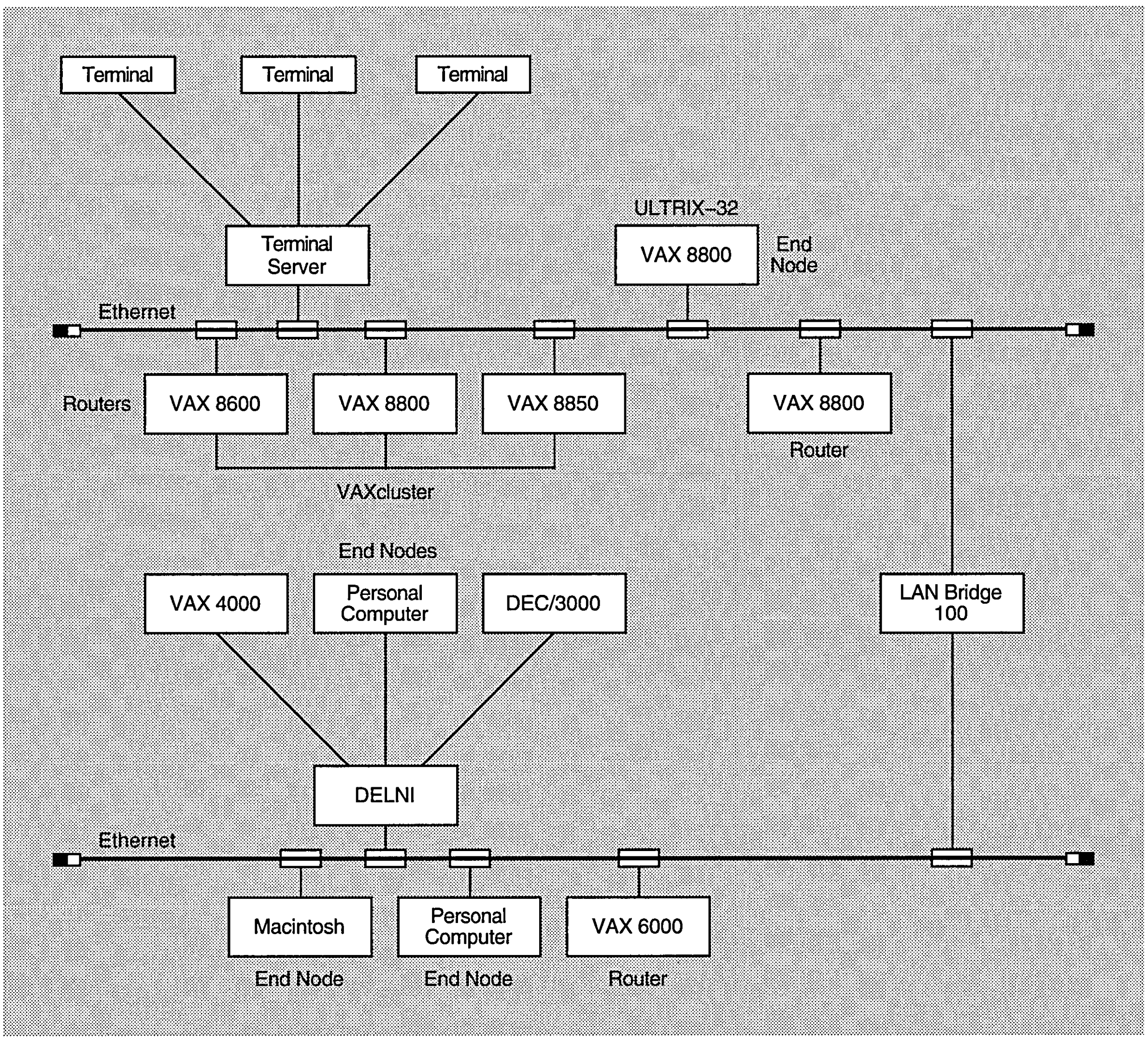
A DECnet network has built-in flexibility in topology and performance. Its architecture adheres to industry standards, and is designed to permit easy expansion and incorporation of new developments in data communications. DECnet offers the option of communicating over different kinds of network connections, which are, for the most part, transparent to the general user of the network.

You can divide very large DECnet networks into multiple areas: up to 63 areas, each containing a maximum of 1023 nodes. In a multiple-area network, nodes are grouped into separate areas, with each area functioning as a subnetwork. Nodes in any area can communicate with nodes in other areas.

Figure 20–3 is an example of a large local area DECnet configuration that illustrates a variety of ways in which you can connect OpenVMS systems to the network. The figure indicates whether a particular node is a router or an end node.

Figure 20–3 shows a larger local area network (LAN) configuration in which two Ethernets are connected by a LAN bridge. Various kinds of operating systems, including nodes in a cluster, are connected directly to the Ethernet. In the figure, a group of small systems is connected to the Ethernet by means of a DELNI. Individual terminal users can gain access to Ethernet nodes through a terminal server.

Figure 20–3 Large Local Area Network Configuration



ZK–6358–GE

20.1.1 How an OpenVMS System Can Be Part of a Network

As the OpenVMS network interface, DECnet supports both the protocols necessary for communicating over the network and the functions necessary for configuring, controlling, and monitoring the network.

You can configure DECnet networking software on any OpenVMS operating system. A DECnet node can communicate with the following:

- Other DECnet nodes in the network
- Nodes with any other operating system that supports DECnet

VAX

- On VAX systems, nodes on other networks, by means of packet-switching networks
- On VAX systems, nodes with foreign vendor systems, by means of gateways, bridges, and other special software and hardware products ♦

DECnet is completely integrated into the OpenVMS operating system; it provides a natural extension of local I/O operations to remote systems. Users can use the network almost transparently. Implementing network applications is straightforward, and network operations are efficient.

You can use DECnet on a standalone node—to run application programs that communicate directly with each other at the task level, for example.

20.1.2 How Nodes Are Connected to the Network

DECnet for OpenVMS supports a variety of network connections, permitting you to link computers and terminals in flexible configurations. The type you use depends on the type of network connection you make: local area, wide area, or worldwide:

- Local area network (LAN) connections

  For local area networks, DECnet supports the the following:

  - Ethernet

    Ethernet, which is shown in Figure 20–1 and Figure 20–2, is a coaxial cable to which each system or device is connected by a single line. In an office or other area where personal computers and workstations are located, ThinWire Ethernet cabling is usually used.

    On the Ethernet, a single, shared network channel LAN, all nodes have equal access. You can add new nodes without affecting existing nodes on the Ethernet. An Ethernet can support up to 1,023 nodes.

  - Fiber Distributed Data Interface (FDDI) LANs

    FDDI LANs provide a reliable, high-speed, multiaccess communications channel. This channel can connect information processing equipment in a limited geographic area, such as an office, a building, or a complex of buildings—a campus, for example.

  VAX

  On VAX systems, nodes in a VAXcluster require DECnet for operating system connections. Each node in a cluster can be connected to an Ethernet that provides the data link for the cluster. If an Ethernet is not available, you can configure the computer interconnect (CI) used by the VAXcluster to be the data link between the cluster nodes. FDDI LANs also support VAXcluster technology and let you configure a computer system with its components spread out over several miles. ♦

- Wide area network (WAN) connections

VAX

On VAX systems, DECnet offers comprehensive wide area network support and long-haul connectivity over point-to-point and multipoint connections:

- Point-to-point connections

  These use the DIGITAL Data Communications Message Protocol (DDCMP) and are synchronous or asynchronous:

  * Synchronous devices provide high-speed connections over local lines or telephone lines (using modems).

  * Asynchronous devices provide low-speed, low-cost connections over terminal lines that are switched on for network use either permanently (a static connection) or temporarily (a dynamic connection). For example, a user at a MicroVAX terminal can configure a dialup line to another computer as a dynamic asynchronous DECnet line for the duration of a call.

- Multipoint connections

  These consist of two or more nodes connected by a synchronous DDCMP communications channel, with one node controlling the channel.

• Worldwide network connections

  DECnet supports worldwide communications with a range of different networks through packet switching networks and gateways. ♦

20.1.3 Understanding the Configuration Database

The system manager at each node in the network is responsible for the DECnet for OpenVMS **configuration database** for the node. Each node in the network has a configuration database, which is stored in the SYS$SYSTEM:NETNODE_REMOTE.DAT file. You can change the location of this file by defining the logical name NETNODE_REMOTE in the SYLOGICALS.COM file. (See Section 5.2.5 for details.)

Besides storing information about other nodes in the network with which the local node can communicate, a configuration database contains the following information:

• Files that describe the following:
  - The local (executor) node
  - The circuits and lines that connect the local node to the network
• Information on the logging collection points (such as the logging monitor) to which network events are reported
• Object databases that describe objects (such as MAIL) known to the network

As system manager, you provide network component information, from the point of view of the local node, in the configuration database at the local node. You can use the Network Control Program (NCP) to build the network configuration database manually or to modify its contents. If you are configuring a node for the first time, you can use the automatic configuration command procedure, NETCONFIG.COM, to establish parameters needed to get DECnet running.

The configuration database is made up of a **volatile database** and a **permanent database**. Table 20–2 describes the two types of databases in the configuration database, compares the duration of changes you make to each, and specifies the NCP commands you use to specify database contents.

Table 20–2 Comparison of Volatile and Permanent Databases

Type of Database	Description	Effect of Modifications	NCP Commands You Can Use
Volatile database	Working copy of the database that reflects current network conditions	Changes exist only while the network is running	Use SET commands to specify the contents of the volatile database. Use CLEAR commands to delete or reset volatile database entries. OPER privilege is required to change a volatile database.
Permanent database	Provides the initial values for the volatile database when you start the network	Changes remain after the network is shut down, but do not affect the current system	Use DEFINE commands to establish the contents of the permanent database. Use PURGE commands to delete permanent database entries. SYSPRV privilege is required to change a permanent database.

20.1.4 How Your System Becomes a Node in the Network

As manager of a DECnet node, you are responsible for establishing your operating system as a node in the network. Following are the steps you need to take. The sections that follow explain these steps in more detail. For specific instructions for performing each step, see the *DECnet for OpenVMS Guide to Networking.*

1. Prepare your system, which includes:
   a. Connecting the hardware.
   b. Planning how you want to configure your system.
2. Make necessary purchases and registrations, including:
   a. Purchasing a DECnet for OpenVMS license and Product Authorization Kit (PAK).
   b. Using the License Management utility to register the PAK.
3. Configure your node in the network, which includes:
   a. Configuring your network environment automatically or manually.

   VAX

   b. On VAX systems, optionally establishing asynchronous connections to other systems. ♦
   c. Verifying that your node is connected to the network.
   d. Providing security for your node.

20.2 Preparations for Joining a Network

This section outlines the preparations you need to make to connect your system to an existing network. Specific instructions for performing these operations are in the *DECnet for OpenVMS Networking Manual.*

Operation	Description
Connect the hardware	To join the network and communicate with other systems, your system must have communications lines. (A communications line connects your computer to the DECnet network.)
Plan the configuration of your node	Planning the configuration of your node in an existing network usually involves coordinating with the system managers of other nodes in the network or with the manager of the network to ensure uniform assumptions about network parameter settings.
Purchase licenses and register a PAK	Before you can bring up your system as a node in the network, you must have a DECnet license and register a DECnet PAK on your system
Configure your node in the network	You can configure the node manually or automatically. You use the manual procedure if you want to modify an existing configuration. You use the automatic configuration procedure, SYS$MANAGER:NETCONFIG.COM, when you first join the network or when you reconfigure your node completely.
Verify your successful connection to the network	To verify your connection to the network, you can perform a number of tests that demonstrate whether your node can communicate with an adjacent node—that is, a node connected to your node by a single physical line. You can also use the DECnet Test Receiver/DECnet Test Sender (DTR/DTS) utility to test this connection.

20.3 Providing Security for Your Node

As manager of a network node, you can protect your system against unauthorized access by users on other nodes in the network by setting passwords for any accounts you create. You can also use the following security measures:

- Protect files and use proxy accounts

  You use the DCL command SET PROTECTION to set limits on who can access the files in your account. If your file is protected, a user on a remote node must be able to specify the user name and password of a local account that has the appropriate privileges to access the file.

  You can permit selected outside users to access particular accounts on your system without sending any explicit access control information over the network. You do this by creating a proxy account that allows a remote user to have access privileges on your node without having a private account on your node.

- Control access to your node

  You can control access to the local node on two levels:

  - Node level

    To control the establishment of logical links with remote nodes, you can specify parameters in your network database access control; these parameters indicate which of the following logical links connections are permitted: INCOMING, OUTGOING, BOTH, or NONE.

To exclude unknown nodes, set Executor Access, which controls the default, to NONE.

- System level

  When a remote user requests access to an object on the local node, a number of means of authorization are checked, including the following:

  * Is an explicit access control string available?
  * Does the user have a proxy account on the local node?
  * Is there a default access account for the object at the local node?
  * Is there a default nonprivileged DECnet account on the local node?

VAX

On VAX systems, you can also control access to the local node by using circuit-level access control. ♦

20.4 Managing a Network Node

Managing a network node usually requires regular monitoring to detect patterns of usage and error conditions on the network, and performing remote configuration of the network to control traffic patterns and accommodate network growth. You can perform maintenance procedures to prevent serious problems from developing, and troubleshooting procedures to resolve problems quickly.

The following sections briefly describe host services you might need to perform, software tools you can use to monitor and manage your DECnet network node, and instructions for shutting down and restarting the network. Refer to the *DECnet for OpenVMS Guide to Networking* for instructions for using these tools and the *DECnet for OpenVMS Networking Manual* for complete information on maintaining, controlling, testing, shutting down, and restarting the network.

20.4.1 Providing Host Services

As manager of a network node, you might also be called upon to provide DECnet host services for other nodes. Host services include:

- Loading system images and programs downline to unattended remote nodes
- Receiving for interpretation upline dumps of system images from nodes that have crashed

For example, DECnet permits you to load an operating system image or a terminal server image downline to a target node. Another DECnet host service involves connecting to an unattended remote node (for example, a diskless communications server) to act as its console.

20.4.2 Monitoring the Network

Using network tools, you can obtain statistics on network usage and routing parameters. Network logging files provide error statistics useful in diagnosing potential problems. NCP commands display the status of nodes, lines, and circuits in the network.

After collecting information about network activity, you can analyze the data you collect to determine whether the network is running properly and whether you need to make changes to resolve problems or improve performance. Table 20–3 shows some of the ways you can monitor the network.

Table 20–3 Ways to Monitor the Network

Method	Use
NCP display commands	To determine the status and characteristics of components in the network
NCP counters	To obtain error and performance statistics on current network operations
Network events logged by DECnet	To report events to you as they happen
Other software tools, such as Ethernet configurator and the DECnet Test Receiver/DECnet Test Sender (DTR/DTS) utility	To learn more about network operation

20.4.2.1 Using NCP Display Commands

You can use the NCP commands SHOW and LIST to monitor network activity:

Command	Description
SHOW	These commands show current condition of network components (no privileges required). Use these commands to monitor operation of the running network.
LIST	These commands list startup values assigned to network components (SYSPRV privilege required).

Table 20–4 shows some of the specific SHOW and LIST commands you can use and the information they display.

Table 20–4 NCP SHOW and LIST Commands

Command	Information Displayed
CHARACTERISTICS	Static information that does not normally change during network operations, such as the identification of the local node
COUNTERS	Counter information about circuits, lines, remote nodes, and the local node
EVENTS	Which network events are currently being logged to which logging collection point
LOGGING	Range of network events being logged by the DECnet Event Logging facility
STATUS	Dynamic information that usually indicates network operation for the running network, such as operational state of the local node
SUMMARY	Only the most useful information from both static and dynamic sources (the default)

20.4.2.2 NCP Counters

You can use NCP command to display error and performance statistics about network components; you can do this at any time while the network is running. DECnet software uses counters to collect statistics automatically for the following:

- Executor node
- Remote nodes

- Circuits
- Lines

To display the contents of counters, you use NCP SHOW COUNTER commands. Following are typical examples of the commands:

```
$ RUN SYS$SYSTEM:NCP
NCP> SHOW EXECUTOR COUNTERS
NCP> SHOW NODE node-id COUNTERS
NCP> SHOW KNOWN CIRCUITS COUNTERS
NCP> SHOW KNOWN LINES COUNTERS
NCP> SHOW LINE line-id COUNTERS
NCP> EXIT
```

For the local node and remote nodes, counter statistics cover connection requests, user data traffic, timeouts, and errors. Specialized counters cover the following:

- Circuit counters: transmission of data packets over the circuit, timeouts, and errors.
- Line counters: transmission of bytes and data blocks over the line and relevant errors.

For a detailed explanation of NCP counters, see the *DECnet for OpenVMS Guide to Networking*. For a complete summary description of all network counters, including the probable causes of particular types of occurrences, refer to the *DECnet for OpenVMS Network Management Utilities*.

20.4.2.3 Using DECnet Event Logging

You can use the DECnet Event Logging facility to monitor important network events, including:

- Changes in circuit and line states (for example, a circuit failure)
- A node becoming reachable or unreachable
- Circuit and node counter values, logged before the counter is automatically set to 0
- Errors in data transmission
- User of invalid data link passwords

20.4.2.4 Using Other Software Tools

Table 20–5 shows some of the additional software tools that are available to view network activity or exercise network operations.

Table 20–5 Network Monitoring Tools

Tool	Description
NCP Ethernet configurator	Permits you to obtain a list of all systems on an Ethernet circuit or circuits
DECnet Test Receiver/ DECnet Test Sender (DTR/ DTS)	Cooperating tasks that perform various functions to exercise network task-to-task capabilities
Monitor utility	Monitors DECnet, displaying information about the use of system resources

20.4.3 Testing the Network

You can use the NCP utility to perform a series of tests to help determine whether the network is operating properly. These tests, which are called **loopback tests**, repeatedly send data through various network components that return the data to its source. If data is not looped successfully, or if the data is returned in a corrupted state, an NCP display indicates that the test failed; the display includes the reasons for the failure and the number of data messages not looped.

You can perform loopback tests at two levels:

Level	Description
Node	These loopback tests check the operation of logical links, routing, and other software.
Circuit	These loopback tests evaluate the operation of circuits. (You cannot perform these tests on asynchronous circuits or lines.)

20.4.4 Shutting Down and Restarting the Network

The network shuts down automatically as part of the system shutdown procedure. If your system is running, you can shut down the network at your local node without destroying any active logical links in one of two ways:

- Shutting down without terminating logical links

  The following command allows no new logical links; when all existing links are disconnected, the network is turned off:

  ```
  $ RUN SYS$SYSTEM:NCP
  NCP> SET EXECUTOR STATE SHUT
  NCP> EXIT
  ```

- Terminating logical links when shutting down

  The following command immediately terminates all logical links and stops the network:

  ```
  $ RUN SYS$SYSTEM:NCP
  NCP> SET EXECUTOR STATE OFF
  NCP> EXIT
  ```

To start the network if it is not currently active, you must log in to the SYSTEM account or have the privileges listed at the beginning of the STARTNET.COM command procedures.

To turn on the network manually, specify the following:

```
$ @SYS$MANAGER:STARTNET
```

Enable the same command in the site-specific startup procedure to have the network start each time the operating system is booted. To enable the command, use a text editor to delete the exclamation point at the start of the command line in the command procedure.

After you enable the command in the startup procedure, the network is turned on automatically as part of the system startup. You do not need to turn on the network again unless you explicitly shut down the network or remove the network startup line from the site-specific startup procedure.

21

Managing InfoServer Systems

This chapter describes InfoServer functions and InfoServer Client for OpenVMS software, which enables OpenVMS systems to access InfoServer device services. The chapter also describes the tasks you must perform to start the client software on your system and to make InfoServer devices available as public devices.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Establishing a server management session	Section 21.3
Starting InfoServer Client for OpenVMS software automatically	Section 21.5.3
Making InfoServer devices available automatically	Section 21.6.3

This chapter explains the following concepts:

Concept	Section
InfoServer functions	Section 21.1
LASTport protocols	Section 21.2
InfoServer Client for OpenVMS functions	Section 21.4
LASTCP utility functions	Section 21.5
LADCP utility functions	Section 21.6

21.1 Understanding InfoServer Functions

The InfoServer system is an Ethernet-based, high-performance, **virtual device server**. It can serve physical device media and sets of logical disk blocks to client systems in a local area network (LAN). Systems running the appropriate client software can connect to virtual devices served by the InfoServer system and use them as though they are locally attached devices.

The InfoServer system is a virtual device server. Unlike a **file server**, the InfoServer system does not impose a file system on the virtual devices that it serves. This means that the InfoServer system can serve disks with any type of on-disk structure. Because the client system itself interprets the on-disk structure, each client can use its own native file system. Multiple on-disk structures can be served by and accessed on a single InfoServer system at the same time.

The InfoServer system can perform the following functions:

- Make compact discs available to clients on the network

  The InfoServer system serves compact discs automatically, using their volume label as the service name when the server is booted or when compact discs are inserted into InfoServer drives. You do not have to perform any management action. Client systems simply bind to and mount the compact discs under their volume labels.

  The InfoServer system can automatically serve to OpenVMS clients compact discs that are in ODS-2 format. High Sierra and ISO-9660 compact discs and other media types can be served manually through the InfoServer management interface.

- Make SCSI tapes available to clients on the network

  The InfoServer system can serve SCSI tape devices to the network using service names. Client systems can connect to these tape devices and use them as though they were locally attached devices.

- Serve read/write disk partitions

  A **partition** is a logical subset of a read/write disk. A single disk can be subdivided into several partitions, each of which can be served to the network independently. To remote client systems, these partitions appear to be whole disks. For example, a client system using InfoServer Client for OpenVMS software can access InfoServer partitions and use them as though they are local hard disks.

- Act as an initial load system for OpenVMS systems

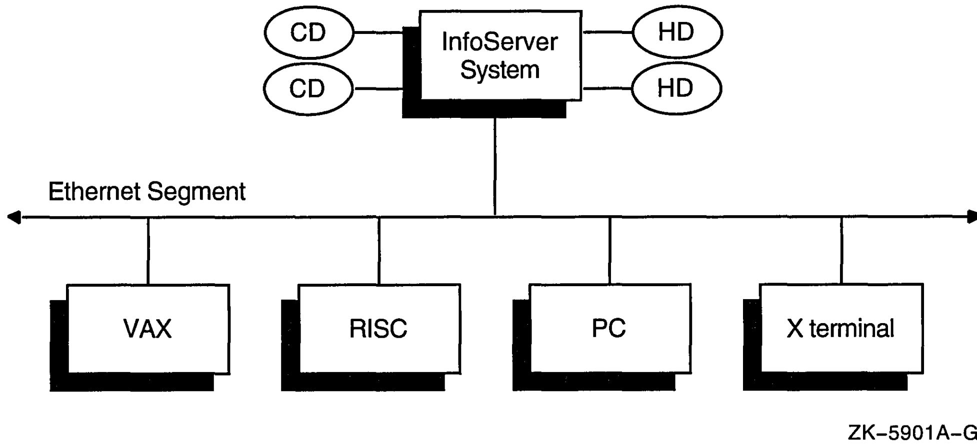
  The InfoServer system can downline load the primary bootstrap program to OpenVMS systems by responding to maintenance operation protocol (MOP) requests. The Initial System Load (ISL) bootstrap program connects back to the OpenVMS software distribution compact disc and boots standalone Backup. The Backup utility is then used to copy the OpenVMS operating system save sets from the compact disc onto a read/write disk attached to the system. All subsequent OpenVMS boots are done from the local read/write disk. See the *InfoServer System Operations Guide* for information on downline loading.

- Downline load other products

  The InfoServer system can be used to load any Ethernet product to a client system that requests a particular file name; that is, the product does not require an NCP database entry to locate the required file. For example, X terminal clients use the InfoServer system to downline load their system software. Special MOP partitions can be created, and the desired file copied to that partition. Each InfoServer system can handle up to 100 simultaneous downline loads more efficiently than host-based downline loaders, which must start processes to assist in the load.

Figure 21–1 shows the relationship of the InfoServer system to several possible client systems. In this figure, two compact discs and two hard disks connected to the server appear to the client systems as local devices. The VAX system and the RISC workstation might be using one or two of the compact discs for software distribution and online documentation, while the PC might be referencing a disk partition on the InfoServer system. The X terminal boots from the InfoServer system and uses InfoServer disks for page, font, and customization files.

Figure 21–1 InfoServer System Serving Clients



You can simply connect the InfoServer system to your Ethernet LAN and turn the system on. After the server is initialized, or **bootstrapped**, the server software automatically makes available, or **serves**, to client systems the device media connected to it. If you insert a compact disc into a server drive, the server detects this new device and automatically serves it to client systems by using the volume label as the service name.

The server bootstraps from its internal read/write device, on which the InfoServer software is preinstalled. InfoServer software updates are distributed on compact discs. As these new releases become available, you can install the software onto the internal device for subsequent booting.

You might want to customize server features. You can control InfoServer functions by logging in to the server and entering server commands, described in the *InfoServer System Operations Guide*.

21.1.1 Automatic Service Policies for Multiple Servers

The InfoServer system automatically serves its locally connected devices to clients when the server is first powered on or when a removable device (for example, a compact disc) is inserted into a drive. The server reads the volume label of each device and uses the label as the name of the service offered to clients.

Note

You can disable the automatic service feature by using the SET SERVER AUTOMOUNT command.

If multiple servers offer the same services, the client uses a rating scheme to select the appropriate service. See the CREATE SERVICE command description in the *InfoServer System Operations Guide* for more information.

When you remove a compact disc from a server disk drive, the InfoServer system ends all client connections to the associated service. The InfoServer system also stops offering, or unserves, the associated service to client systems.

21.1.2 High-Availability Feature to Reduce Service Interruptions

The InfoServer system provides a high-availability feature that is especially beneficial for OpenVMS clients. If the server ends a service connection for some reason (for example, the server reboots, or you remove a compact disc), the OpenVMS client enters mount verification for that volume. If the same service is offered elsewhere on the network, the client automatically connects to the other volume.

For example, suppose you have two identical copies of the OpenVMS Online Documentation compact disc in drives on two different servers. If the original service is broken, the connection fails over to the second compact disc. File operations continue as normal, and users experience almost no service disruption.

21.1.3 Support for X Terminal Clients

X terminal clients use the InfoServer system to download their system software, provide font services, save configuration information, and page memory to and from InfoServer disks. For example, system files for Digital's VXT 2000 windowing terminals can be installed from compact disc on the InfoServer system. Once installed, these files are downline loaded on demand to each terminal when it is powered on.

The terminals can dynamically allocate partitions on an InfoServer disk as needed. For example, when a user requests that terminal customizations be saved, the InfoServer system automatically creates a disk partition to hold the information and creates a network service name for that partition. Once customization information is saved, the user can recall the information at any time.

VXT 2000 terminals that are InfoServer clients can also be virtual memory machines. Such terminals can page sections of main memory to and from InfoServer disks as required. Because a VXT client has no local disk, it uses InfoServer disks as page disks. When main memory needs to be paged out to disk, the VXT client requests the InfoServer system to create a partition. This partition is then automatically extended as needed. Partitions and their network service names are created dynamically, without the need for any user action.

By default, the InfoServer disk DK1, which is the internal disk that ships with each InfoServer 150 system, is enabled to allow VXT 2000 clients to allocate partitions remotely. Other disks can also be enabled through the use of InfoServer commands.

21.2 Understanding LASTport Protocols

The InfoServer system uses the LASTport transport protocol and the LASTport/Disk and LASTport/Tape system application protocols to provide access to the virtual devices it serves to the LAN. These protocols provide high-performance access to disk and tape devices. The InfoServer system implements the server portion of the protocols, while the client systems that access InfoServer storage devices implement the client portion.

21.2.1 LASTport Transport Protocol

The LASTport protocol is a specialized LAN transport protocol that allows many clients to access InfoServer systems and perform reliable transactions. For the InfoServer system, a transaction is a device read or write operation. The LASTport protocol allows many client systems concurrently to read information from and and write information to an InfoServer storage device.

Unlike timer-based protocols, the LASTport protocol is a transaction-oriented protocol. Normally, information does not pass between a client and an InfoServer system unless the client initiates a transaction. The client system then runs a timer on the transaction, normally waiting from two to five seconds before assuming that the transaction is lost and retrying the operation.

The LASTport protocol does not provide any routing functions; it runs only in a LAN. The LASTport protocol type is 80–41. If the extended LAN uses any filtering devices, they must allow this protocol type to pass unfiltered so that clients can access InfoServer systems across the filtering device.

The InfoServer system uses a multicast address feature of the LASTport protocol to establish connections to devices. The format of the multicast address is 09–00–2B–04–*nn–nn*, where *nn* depends on the work group enabled (see the *InfoServer System Operations Guide*).

21.2.2 LASTport/Disk Protocol

The LASTport/Disk protocol is a specialized disk protocol that uses the LASTport transport. That is, LASTport/Disk messages are delivered in LASTport messages. The LASTport/Disk protocol provides the mechanism for reading and writing logical disk blocks independent from any underlying file system. The clients that implement the LASTport/Disk protocol interpret the file system locally. By using the LASTport/Disk protocol for disk access, the InfoServer system can support multiple client operating systems and on-disk structures concurrently.

The LASTport/Disk protocol also provides the naming facility to access disks. The InfoServer system assigns each virtual disk a service name and allows clients to query the LAN for these names. When the requested service is found, the client connects to it, and disk access can begin. When duplicate virtual disks are available under identical service names, the protocol provides a facility for load balancing among the available disks.

21.2.3 LASTport/Tape Protocol

Like the LASTport/Disk protocol, the LASTport/Tape protocol uses the LASTport transport. That is, LASTport/Tape messages are delivered in LASTport messages. The LASTport/Tape protocol provides the mechanism for reading and writing tape records. Tape devices attached to the InfoServer system appear to tape clients as locally attached devices.

The LASTport/Tape protocol also provides the naming facility to access tapes. The InfoServer system assigns each tape device a service name and allows clients to query the LAN for these names. When the requested service is found, the client connects to it, and tape access can begin.

21.3 Establishing a Server Management Session

You can establish a server management session from a local or remote console terminal:

- **For a local session,** you connect a terminal capable of interpreting VT100 ANSI escape sequences to the MMJ serial port 1 on the rear of the InfoServer system unit. The terminal must be set to 9600 baud, 8 bits, no parity.
- **For a remote session,** you make a connection to the InfoServer system through a local area terminal (LAT) server.

  Like many network servers, the InfoServer system advertises a LAT service for its management interface and accepts connections from remote terminals attached to terminal servers. Therefore, any terminal attached to a terminal server on the extended LAN can act as a console terminal for the InfoServer system (if the user knows the InfoServer management password).

Determining the Server's Default Service Name

To make a remote connection to the InfoServer system for the first time, you must first determine the server's default name. To do this, add the four-character prefix LAD_ to the hexadecimal Ethernet datalink address on the InfoServer system's cabinet. You can change this default name by using the InfoServer command SET SERVER NAME.

The server's name is the LAT service name to which you connect. For example, if the default server name is LAD_08002B15009F, then you would enter the following command at the terminal server's prompt to manage the InfoServer system:

```
Local> CONNECT LAD_08002B15009F
```

See your terminal server user's guide for information about the establishment of LAT service connections.

Entering an InfoServer Password

After you connect to the InfoServer system, you must enter an InfoServer password to establish the management session. The default server password is ESS. You can change the password with the InfoServer command SET SERVER PASSWORD.

Example

The following example shows the establishment of a sample session using a DECserver 500 terminal server:

```
Local> CONNECT LAD_08002B133C1C
Password: ESS (not echoed)
Local -010- Session 1 to LAD_08002B133C1C established

DEC InfoServer V2.2

InfoServer> SHOW SERVER
```

In this example, the terminal server's prompt is Local>, and a LAT session is established to the InfoServer system whose service name is LAD_08002B133C1C. The InfoServer system prompts for a server password. When you enter the correct password, the server prompts for InfoServer commands with the InfoServer> prompt.

Ending a Session

At the end of the management session, you can enter the EXIT command at the InfoServer> prompt. This command returns you to the terminal server's Local> prompt if the management session is over a LAT connection.

21.3.1 Server Management Commands

Table 21–1 summarizes InfoServer commands and their functions.

Table 21–1 Summary of InfoServer Commands

Command	Function
BACKUP	Saves InfoServer format disks.
CLEAR	Erases the screen of the terminal running the management session.
COPY	Copies data from one disk or partition to another disk or partition.
CRASH	Causes the server software to take a recognizable bugcheck, creating a dump if crashdump processing is enabled.
CREATE	Creates a new partition on a read/write device or creates a new service.
DELETE	Deletes a partition or service that was previously created.
EXIT	Terminates the management session.
HELP	Displays help text for the InfoServer commands.
INITIALIZE	Formats a read/write disk into an InfoServer disk.
LOOP	Automatically repeats any valid InfoServer command.
MONITOR	Automatically repeats valid InfoServer commands every 3 seconds, clearing the screen and placing the cursor at the home position.
PURGE	Purges old versions of VXT software.
REBOOT	Shuts down and reboots the server.
RESTORE	Resets the server to a previously saved system configuration.
RETRIEVE	Restores InfoServer format disks saved by the BACKUP command.
REWIND	Rewinds an InfoServer tape.
SAVE	Saves configuration and service data for recovery after a server reboot.
SET	Sets partition, service, or server parameters.
SHOW	Displays the server's parameters and counters.
UNLOAD	Rewinds and unloads an InfoServer tape.
UPDATE	Installs one or more new software products or functions.
VERIFY	Validates the on-disk structure of a device formatted with the INITIALIZE command.
ZERO	Sets internal server counters to 0.

The InfoServer system provides a Help facility that contains information about each server command, including parameters, qualifiers, and examples of its use. For detailed information on InfoServer commands, refer to the *InfoServer System Operations Guide*.

21.4 Understanding InfoServer Client for OpenVMS Functions

InfoServer Client for OpenVMS software enables clients running the OpenVMS operating system to access virtual device services offered by InfoServer systems on a LAN. Software components include the following:

- LASTport driver

  The LASTport driver provides reliable data transfer services for its clients. It interacts with the Data Link driver and the LASTport/Disk driver as an efficient transport for a virtual device service. The LASTport driver can support other applications, such as a primitive data queueing service.

- LASTport/Disk client driver

  The LASTport/Disk client driver presents a standard block device interface to the system. The OpenVMS file system interacts with the LASTport/Disk client as if the LASTport/Disk client is a local disk driver. The LASTport/Disk client driver supports both raw and buffered interfaces.

- LASTport/Tape client driver

  The LASTport/Tape client driver enables OpenVMS clients to access and use as local devices SCSI tapes attached to InfoServer systems.

- LASTCP and LADCP utilities

  These utilities allow you to start InfoServer Client software on your system, monitor transport status, and configure and maintain InfoServer device services. Section 21.5 and Section 21.6 introduce the utilities. For complete information about the utilities, refer to the *InfoServer Client for OpenVMS LASTCP and LADCP Utilities* manual.

21.5 Understanding LASTCP Utility Functions

InfoServer Client for OpenVMS software uses the LASTport protocol to communicate with InfoServer systems on an extended LAN. The protocol is implemented in the OpenVMS device driver ESS$LASTDRIVER.

The LASTCP utility is the management interface that allows you to control and diagnose ESS$LASTDRIVER. You can use LASTCP to do the following:

- Start and stop ESS$LASTDRIVER
- Display counters for circuits, lines, nodes, and ESS$LASTDRIVER
- Display node characteristics
- Display known clients and servers
- Display LASTport status
- Reset counters

The description of the LASTCP utility covers the following topics:

- Invoking and exiting the utility
- Starting InfoServer Client for OpenVMS software automatically
- LASTCP command summary

21.5.1 Invoking and Exiting the Utility

Use of LASTCP requires normal privileges, except where noted. To invoke LASTCP, enter the following command:

```
$ RUN SYS$SYSTEM:LASTCP
%LASTCP-I-VERSION, ESS$LASTDRIVER V1.5 is running
LASTCP>
```

At the LASTCP> prompt, you can enter LASTCP commands. To exit the utility, enter EXIT or press Ctrl/Z after the LASTCP> prompt.

You can also execute a single LASTCP command by using a DCL string assignment statement, as shown in the following example:

```
$ LASTCP :== $LASTCP
$ LASTCP SHOW CLIENTS
```

LASTCP executes the SHOW CLIENTS command and returns control to DCL command level.

21.5.2 LASTCP Command Summary

Table 21–2 summarizes LASTCP commands and their functions.

Table 21–2 Summary of LASTCP Commands

Command	Function
EXIT	Returns the user to DCL command level
HELP	Displays HELP text for LASTCP commands
SHOW CIRCUIT COUNTERS	Displays circuit counters
SHOW CLIENTS	Displays known clients
SHOW LINE COUNTERS	Displays line counters
SHOW NODE CHARACTERISTICS	Displays node characteristics
SHOW NODE COUNTERS	Displays node counters
SHOW SERVERS	Displays known servers
SHOW STATUS	Displays local status
SHOW TRANSPORT COUNTERS	Displays transport counters
START TRANSPORT	Starts LASTDRIVER
STOP TRANSPORT	Stops LASTDRIVER
ZERO COUNTERS	Resets counters

You can abbreviate LASTCP commands to the first unique characters of the command verb. For example, you can abbreviate the command SHOW SERVERS to SH SE.

LASTCP provides a Help facility that contains information about each command and its parameters and qualifiers, as well as examples of its use. For a complete description of LASTCP commands, refer to the *InfoServer Client for OpenVMS LASTCP and LADCP Utilities* manual.

21.5.3 Starting InfoServer Client for OpenVMS Software Automatically

You must start InfoServer Client for OpenVMS software using the ESS$STARTUP command procedure. To make sure the software is started automatically each time the system reboots, execute the startup procedure from within SYSTARTUP_VMS.COM.

How to Perform This Task

1. Determine the value of SCSNODE, your system's node name parameter. If the parameter is defined as the null string (the default value), InfoServer Client for OpenVMS software does not start.

   If you are running or plan to run DECnet for OpenVMS, SCSNODE must be defined as the system's DECnet node name. If you do not plan to run DECnet, and if the system is a VMScluster member, SCSNODE must be defined as the SCS system name, a 1- to 8-character node name that is unique in the cluster.

   To determine the value of SCSNODE, enter the following commands to invoke SYSMAN and display the parameter:

   ```
   $ RUN SYS$SYSTEM:SYSMAN
   SYSMAN> PARAMETERS USE CURRENT
   SYSMAN> PARAMETERS SHOW SCSNODE
   ```

2. If SCSNODE is defined as the null string, perform these steps:

   a. Enter a command in the following format, where *node-name* is the system's DECnet node name or (if you do not plan to run DECnet for OpenVMS) the SCS system name:

      PARAMETERS SET SCSNODE "node-name"

      For example:

      ```
      SYSMAN> PARAMETERS SET SCSNODE "MYNODE"
      ```

   b. Enter the following commands to write the new value to the parameter file and exit from SYSMAN:

      ```
      SYSMAN> PARAMETERS WRITE CURRENT
      SYSMAN> EXIT
      ```

   c. Add a line in the following format to the AUTOGEN parameter file SYS$SYSTEM:MODPARAMS.DAT to define the SCSNODE parameter:

      SCSNODE = "node-name"

      For example:

      ```
      SCSNODE = "MYNODE"
      ```

3. Invoke any editor to edit SYS$MANAGER:SYSTARTUP_VMS.COM and find the command that starts InfoServer Client software. For example:

   ```
   $ @SYS$STARTUP:ESS$STARTUP DISK
   ```

   Note that the parameters CLIENT and DISK are synonymous. If the command is preceded by the DCL comment delimiter (!), remove the delimiter. If you want to enable tape functions, add the TAPE parameter to the command line:

   ```
   $ @SYS$STARTUP:ESS$STARTUP DISK TAPE
   ```

4. If SYSTARTUP_VMS.COM invokes the DECnet for OpenVMS startup procedure (SYS$MANAGER:STARTNET.COM), make sure SYSTARTUP_VMS.COM executes the InfoServer Client for OpenVMS startup procedure *after* it invokes STARTNET.COM.

   The following example shows the network startup command line followed by the InfoServer Client for OpenVMS startup command line. Note that if you omit the TAPE parameter, only the disk function is started.

   ```
   $ @SYS$MANAGER:STARTNET
      .
      .
      .
   $ @SYS$STARTUP:ESS$STARTUP DISK TAPE
   ```

5. Optionally, edit the file SYS$STARTUP:ESS$LAST_STARTUP.DAT to specify desired startup qualifiers for the LASTport transport. (See the *InfoServer Client for OpenVMS LASTCP and LADCP Utilities* manual.)

21.5.4 Startup Restrictions: PATHWORKS and RSM

If PATHWORKS or Remote System Manager (RSM) or both are installed, the InfoServer Client for OpenVMS startup must be run before the startup for PATHWORKS or RSM, or both.

```
$ @SYS$MANAGER:STARTNET
   .
   .
   .
$ @SYS$STARTUP:ESS$STARTUP DISK TAPE
$ @SYS$STARTUP:PCFS_STARTUP
$ @SYS$STARTUP:RSM$SERVER_STARTUP
```

InfoServer Client for OpenVMS software provides device drivers and control programs that are shared by both the PATHWORKS and RSM products. All InfoServer Client for OpenVMS components are prefixed with ESS$. The drivers and control programs supplied with InfoServer Client for OpenVMS software provide all necessary support for both PATHWORKS and RSM in addition to InfoServer Client support. You must execute the InfoServer Client for OpenVMS startup in the site-specific startup before executing either the PATHWORKS or RSM startup procedures.

21.5.5 Startup Restrictions: SYSMAN

You cannot start InfoServer Client for OpenVMS from a subprocess. Because the OpenVMS System Management utility (SYSMAN) uses subprocesses to complete its tasks on remote nodes, SYSMAN cannot be used to execute the SYS$STARTUP:ESS$STARTUP procedure.

21.5.6 User Account Requirements

To work with InfoServer Client for OpenVMS software, user accounts on your system must have the following privileges and quotas:

- You need GRPNAM privilege if the /GROUP qualifier of the LADCP BIND command is used; SYSNAM privilege is required if the /SYSTEM qualifier of the LADCP BIND command is used.
- At a minimum, you need default UAF account quotas.

See the AUTHORIZE section in the *OpenVMS System Management Utilities Reference Manual* for a description of how to verify and change account privileges and quotas.

21.5.7 System Parameter MAXBUF Requirement

To use all the utility's SHOW functions, you must set the value of the system parameter MAXBUF to 32000.

21.6 Understanding LADCP Utility Functions

You use LADCP to configure and control the LASTport/Disk and LASTport/Tape protocols on OpenVMS systems. OpenVMS systems that use LASTport/Disk and LASTport/Tape services are called client systems. You can use LADCP to do the following:

- Establish **bindings** to services. A binding creates a new DAD*n*: virtual disk unit or a new MAD*n*: virtual tape unit on the local OpenVMS system.
- Remove bindings to services.

You can control service access by using a service access password. You can also write-protect services. In this case, local OpenVMS users of a DAD*n*: or MAD*n*: device unit receive an error if they attempt a write operation to the unit.

The protocols allow you to access storage devices that reside on an InfoServer system as though they are locally connected to your OpenVMS system. Thus, several OpenVMS client systems can share the same read-only media, eliminating the need for duplicate drives and media.

DAD*n*: and MAD*n*: device units are also referred to as **virtual device units**. They represent the local OpenVMS context for a volume that resides on a remote server. The OpenVMS driver that controls the DAD*n*: units is called ESS$DADDRIVER. The OpenVMS driver that controls the MAD*n*: units is called ESS$MADDRIVER.

The LASTport/Disk and LASTport/Tape protocols depend on the LASTport transport. The ESS$STARTUP.COM command procedure in SYS$STARTUP automatically loads ESS$DADDRIVER and ESS$MADDRIVER as well as ESS$LASTDRIVER, the LASTport transport driver.

Note

Your site-specific startup command procedure must include a call to ESS$STARTUP.COM. If you are using DECnet software, you must place the call *after* the SYS$MANAGER:STARTNET.COM command that starts DECnet software. See Section 21.5.3.

21.6.1 Invoking and Exiting the LADCP Utility

To invoke LADCP, enter the following command:

```
$ RUN SYS$SYSTEM:ESS$LADCP
LADCP>
```

You can enter LADCP commands at the LADCP> prompt.

You can also execute a single LADCP command by using a DCL string assignment statement, as shown in the following example:

```
$ LADCP :== $ESS$LADCP
$ LADCP BIND CD_DOC_00661 /NOWRITE
```

LADCP executes the BIND command and returns control to DCL command level.

To exit LADCP, enter EXIT or press Ctrl/Z after the LADCP> prompt.

21.6.2 LADCP Command Summary

Table 21–3 summarizes LADCP commands and their functions.

Table 21–3 Summary of LADCP Commands

Command	Function
BIND	Establishes a LAD service binding and creates a device unit.
EXIT	Returns the user to DCL command level.
HELP	Displays help text for LADCP commands.
SHOW SERVICES	Displays services offered by all available InfoServer systems on the network.
UNBIND	Eliminates an established LAD service binding.

LADCP provides a Help facility that contains information about each LADCP command, including parameters, qualifiers, and examples of its use. For detailed descriptions of LADCP commands, refer to the *InfoServer Client for OpenVMS LASTCP and LADCP Utilities* manual.

21.6.3 Making InfoServer Devices Available Automatically

You can make remote InfoServer devices available on your system each time the system boots. To do so, add a series of LADCP BIND commands to SYSTARTUP_VMS.COM. For more information about the BIND command, see the *InfoServer Client for OpenVMS LASTCP and LADCP Utilities* manual.

How to Perform This Task

1. Edit SYSTARTUP_VMS.COM and find the command that starts InfoServer Client software. For example:

   ```
   @SYS$STARTUP:ESS$STARTUP DISK TAPE
   ```

   This command starts the software with disk and tape functions.

2. Add the following command to invoke LADCP:

   ```
   $ RUN SYS$SYSTEM:ESS$LADCP
   ```

3. Immediately following this command, add BIND commands in the following format to make any InfoServer compact discs or hard disks available as virtual device units:

   BIND [/QUALIFIER,...] service-name

   To make tape devices available, you must specify the /TAPE qualifier in addition to any other desired qualifiers:

   BIND/TAPE [/QUALIFIER,...] service-name

   For *service-name*, specify the name of the InfoServer device service. Usually a service name is the label of the volume to which the InfoServer system is providing access. For more information on the BIND command, see the *InfoServer Client for OpenVMS LASTCP and LADCP Utilities* manual.

4. Add an EXIT command to exit LADCP.

5. Add MOUNT commands in the following format to make available as public devices the virtual device units created in the previous step:

   MOUNT/SYSTEM/NOASSIST device-name volume-label

   For *device-name*, specify the name of the device. For *volume-label*, specify a volume label to assign to the device. For more information on the MOUNT command, see the MOUNT section in the *OpenVMS System Management Utilities Reference Manual*.

Example

The following commands, executed in SYSTARTUP_VMS.COM, start the InfoServer Client software and make available the InfoServer device DAD$VMS055.

```
$ @SYS$STARTUP:ESS$STARTUP DISK
$ RUN SYS$SYSTEM:ESS$LADCP
  BIND VMS055
  EXIT
$ MOUNT/SYSTEM/NOASSIST DAD$VMS055 VMS055
```

In this example, the VMS Version 5.5 compact disc distribution kit, loaded into a remote compact disc drive connected to an InfoServer system, is made available on the system as a virtual device unit and mounted as a public device.

22
Managing the LAT Software

This chapter describes how the LAT software works and the tasks you must perform to implement and manage the LAT software on your system.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Starting up the LAT protocol	Section 22.4
Customizing LAT characteristics	Section 22.5
Creating a service	Section 22.5.1
Setting up ports	Section 22.5.2
Setting up printers	Section 22.5.2.1
Setting up special application services	Section 22.5.2.2
Enabling outgoing LAT connections	Section 22.5.3
Managing the LATACP database size	Section 22.6

This chapter explains the following concepts:

Concept	Section
The LAT protocol	Section 22.1
The LAT network	Section 22.2
The LATCP utility	Section 22.3

22.1 Understanding the LAT Protocol

The operating system uses the LAT software to communicate with **terminal servers** and other systems within a local area network (LAN). Terminal servers are communication devices dedicated for connecting terminals, modems, or printers to a LAN. They offer the following features:

- Provide a cost-effective method of connecting many user terminals to a computer
- Save on cable requirements
- Maximize the number of devices that can access a computer

With the LAT software, which implements the LAT **protocol**, the operating system can offer resources, or **services**, that the terminal servers can access. A system that offers LAT services is called a **service node**. In addition, nodes can access LAT services by enabling outgoing connections (using LATCP) and using the DCL command SET HOST/LAT. (In the remainder of this chapter, "servers" refers both to dedicated terminal servers and to nodes that allow outgoing access to other LAT services.)

A LAT service can consist of all the resources of a computer system, or it can be a specific resource on a computer system, such as an application program. You can set up your system as a **general timesharing service**, meaning that all of its resources are available to users in the LAN, or you can restrict access to a specific service (application program) on the system. This chapter and the *OpenVMS I/O User's Reference Manual* outline the procedure you use to set up access to a dedicated application program.

22.1.1 How the LAT Protocol Works

The LAT protocol allows the terminal servers and computers to communicate within a LAN, such as the Ethernet or the Fiber Distributed Data Interconnect (FDDI). The LAT protocol matches terminals and other devices to the computing resources (services) of the LAN. Because LAT terminals are not connected directly to the computer (service node) they are accessing, the local server must listen for service requests from its terminals and be able to match the terminals with computers that provide the desired services.

Using the LAT protocol, then, the operating system announces its available services over the LAN. Servers listen to the LAN announcements and build a database of service information so that they can locate an appropriate system when a user terminal requests computing services. For example, a user terminal might request general processing service or a data entry program on the operating system. A server uses the LAT protocol to establish and maintain a connection between the requesting terminal and the operating system.

Sometimes the operating system can request services from a terminal server. The LAT protocol allows systems to ask for connections to printers or other devices attached to a terminal server.

22.1.2 Advantages of the LAT Protocol

There are many advantages to using the LAT protocol on your system:

- The LAT protocol lets you make the resources of any computer on a local area network available to any user in that network.
- In addition to general processing resources, you can set up terminals, printers, and modems so they are available from multiple systems in the local area network. This lets you efficiently use these resources and keep them available even if one of the systems in the network must be shut down.
- You can also set up application programs, such as data entry programs or news services, as resources. When a user requests a connection to the resource, the LAT protocol sets up a connection directly to the application program. No login procedure is necessary.
- The LAT protocol provides load balancing features and recovery mechanisms so users get the best, most consistent service possible. In their broadcast messages, systems rate the availability of their services so that servers can establish connections to computing resources on the least busy node. If a node becomes unavailable for any reason, the servers attempt to provide access to alternate services.

- Users can establish multiple computing sessions on their terminals, connecting to several different computers and switching easily from one computing session to another. After switching from one session to another, users can return to the previous session and pick up where they left off. This saves users the time normally required to close out and reopen files or accounts and to return to the same point in a session.
- Finally, the LAT protocol can provide improved system performance. Because the servers bundle messages onto a single LAN interface, a server interface decreases the network traffic and reduces the number of computer interrupts realized in systems where terminals, modems, and printers each have a physical connection to the computer.

22.2 Understanding the LAT Network

A **LAT network** is any local area network where terminal servers and operating systems use the LAT protocol. A LAT network can coexist on the same LAN with other protocols. The LAT protocol, which operates on both terminal servers and the operating systems, is designed to ensure the safe transmission of data over the LAN.

The LAT network consists of the following components:

Component	For More Information
Service nodes	Section 22.2.1
Terminal server nodes	Section 22.2.2
Nodes allowing outgoing connections	Section 22.2.3
LAN cable	Section 22.2.4

Service nodes supply computing resources for the local network, while terminal server nodes (or nodes allowing outgoing connections) port their terminals, modems, or printers to those resources upon request from a user terminal or an application program.

You can use the LAT Control Program (LATCP) to configure the LAT characteristics for your system. LATCP allows you to set up your system to support:

- Incoming access only
- Outgoing access only
- Both incoming and outgoing access

The systems that support incoming LAT connections are **service nodes**. (Using LATCP, you can also set up your system so that it supports neither incoming nor outgoing access.)

22.2.1 Service Nodes

A service node is one type of node in a LAT network. (Nodes that are not running an OpenVMS operating system can also be used along with the OpenVMS nodes in a LAT network.) A service node is an individual computer in a LAN that offers its resources to users and devices. Because the OpenVMS operating systems contain the LAT protocol, any OpenVMS system can be configured as a service node within a LAT network.

22.2.1.1 Types of Services

Each node offers its resources as a **service**. Often, a node offers a general processing service, but it can offer special application services as well. Any or all of the services can be specialized applications.

For example, your service node might offer services for the following:

- General processing
- Data entry
- Stock quotations

The general processing service would allow the use of the general computing environment. The data entry and stock services, on the other hand, would be restricted environments, with connections to the application service but to no other part of the service node.

Each service is distinguished by the name the system manager assigns to it. In a VMScluster, Digital recommends that the service name be the same as the cluster name. In an independent node, Digital recommends that the service name be the same as the node name. With special service applications, the service holds the name of the application.

22.2.1.2 Service Announcements

A service node announces its services over the LAN at regular intervals so that terminal servers (and OpenVMS systems that allow outgoing connections) know about the availability of these network resources. The service announcement provides the physical node name, the service names, a description of services, and a rating of service availability. Servers listen to the LAN announcements and record information in a database. On nodes allowing outgoing connections, this database is maintained by the LAT Ancillary Control Process (LATACP). (See Section 22.6 for more information about managing the LATACP database.)

Whenever a user terminal or application program requests a service, the server node connects to the appropriate service node.

22.2.1.3 Print Requests

In some cases, service nodes can request services from terminal servers. The most common situation is when the system wants to use a printer that is connected to a terminal server port. The system submits the print request to the terminal server print queue that is set up and initialized in the OpenVMS startup procedure. Then the LAT symbiont (the process that transfers data to or from mass storage devices) requests the LAT port driver to create and terminate connections to the remote printer.

For information about setting up queues for printers connected to LAT ports, see Section 13.6.4.

22.2.2 Terminal Server Nodes

A **terminal server node** is the second type of node in a LAT network. A terminal server node is usually located near the terminals and printers it supports. The terminals and printers are physically cabled to the terminal server; the terminal server is physically connected to the LAN cable.

22.2.2.1 Locating Service Nodes

Terminal servers build and maintain a directory of services from announcements advertised over the network. Then, when terminal servers receive requests from terminal users, they can scan their service databases and locate the computer that offers the requested service.

Terminal servers not only look for the node that provides the requested service, but they can also evaluate the service rating of that node. If a requested service is offered by more than one node, then the service rating is used to select the node that is least busy. A server establishes a logical connection between the user terminal and the service node.

22.2.2.2 Setting Up Connections

One logical connection carries all the data directed from one terminal server node to a service node. That is, the server combines data from all terminals communicating with the same node onto one connection. A terminal server establishes a logical connection with a service node only if a logical connection does not already exist.

If a connection fails for any reason, a terminal server attempts to find another node offering the same service and "rolls over" the connection so users can continue their computing sessions.

Even though terminal connections are bundled together, each terminal can be uniquely identified by its name. A terminal name consists of two parts: the first part is the name of the port on the terminal server that the terminal line plugs into; the second part is the name of the terminal server node.

22.2.2.3 Servicing Nodes

Although terminal servers are usually the requesting nodes in a LAT network, sometimes service nodes request service from terminal servers. Most commonly, a service node queues print requests to remote printers connected to terminal servers.

22.2.3 Nodes Allowing Outgoing Connections

Nodes can be set up to allow incoming connections, outgoing connections, or both. Nodes (excluding those that offer incoming connections only) such as terminal server nodes can locate service nodes and set up connections. The database of information about available nodes and services is maintained by the LAT Ancillary Control Process (LATACP). (See Section 22.6 for more information about managing the LATACP database.)

On a node that is set up to allow outgoing LAT connections, a user can connect to another node in the LAT network by entering the SET HOST/LAT command. For more information, see the SET HOST/LAT command in the *OpenVMS DCL Dictionary*.

22.2.4 Sample LAT Configuration

Figure 22–1 demonstrates the components of a LAT network. The network consists of an Ethernet cable connecting service nodes and terminal server nodes.

The three service nodes in Figure 22–1, named MOE, LARRY, and ALEXIS, each offer services to terminal server nodes on the network.

Two of the service nodes, MOE and LARRY, belong to the OFFICE cluster. (The cluster is distinguished by its computer interconnect [CI] and star coupler.) Because MOE and LARRY are clustered, their service names are the same as their cluster name. Because both service nodes offer an OFFICE service, terminal server nodes can assess the work load on both OFFICE nodes and establish a connection to a node that offers the service that is less busy.

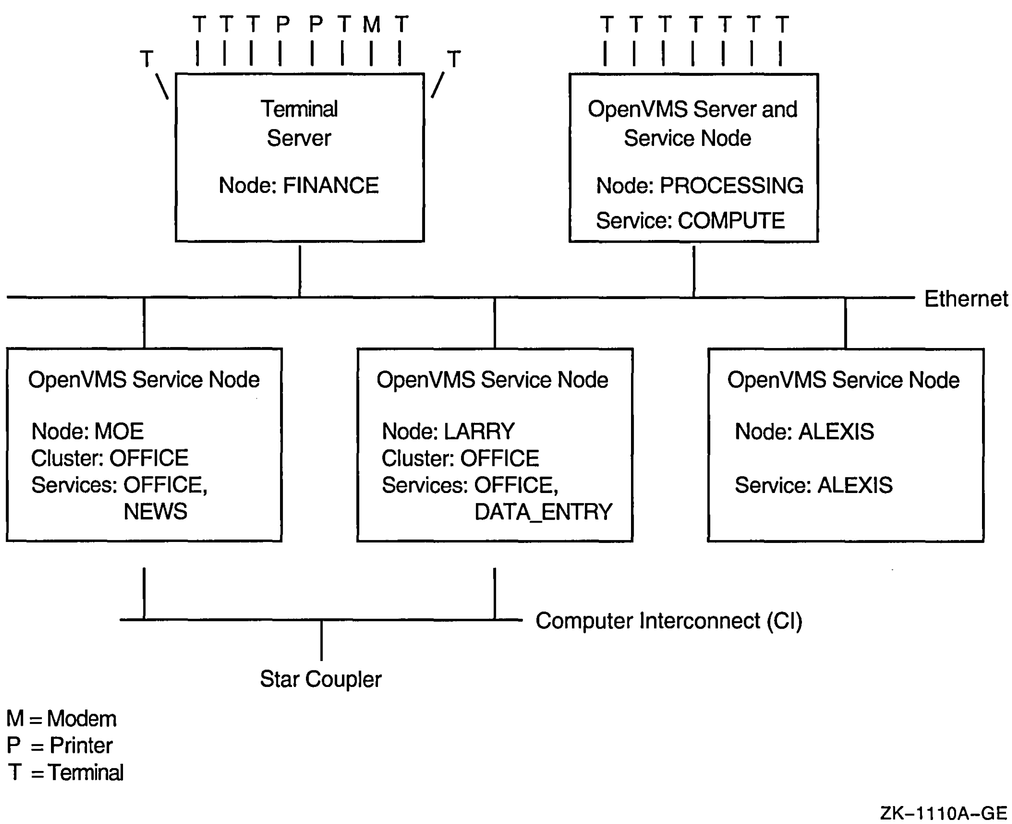
The third service node, ALEXIS, is an independent node in the LAT network so its service name is the same as its node name.

In addition to its primary OFFICE service, node MOE offers an application service called NEWS. With this specialized service, user terminals can connect directly to the online news program, without any login procedure but also without general access to the general computer resources of the node.

The node FINANCE, shown in Figure 22–1, is a terminal server node; it supports a number of interactive terminals, a modem, and a printer. The node PROCESSING is a node allowing outgoing connections; it supports several interactive terminals. The node FINANCE can accept print requests from any of the three service nodes, provided each of the service nodes has set up print queues to support remote printers on the terminal server.

Node PROCESSING is also a service node. It offers the service COMPUTE.

Figure 22–1 Sample LAT Network Configuration



22.2.5 LAT Relationship to VMSclusters and DECnet

Although the LAT protocol works independently of VMScluster software, Digital recommends that you configure a service node to complement the VMScluster concept. You achieve this by creating a service on each node in a VMScluster and assigning the cluster name to this service. A terminal server assesses the availability of cluster services and establishes a connection to the node that is least busy. Thus, the LAT protocol helps balance the cluster load. If one node in the VMScluster fails, the terminal server can transfer the failed connections to another service node within the VMScluster.

The LAT software does not use DECnet as a message transport facility, but instead uses its own virtual circuit layer to implement a transport mechanism. The LAT and DECnet software work independently in a common LAN environment. (Note, however, that the DECnet software must start up before the LAT software.) For compatibility, if a service node is also a DECnet node, the service node name should be the same as the DECnet node name.

22.3 Understanding the LATCP Utility

The LAT Control Program utility (LATCP) is a utility program used for configuring and controlling the LAT software on OpenVMS systems. LATCP commands let you stop and start the LAT driver (which implements the LAT protocol) and modify or display LAT characteristics of the OpenVMS node.

With LATCP, you can set up your system as a service node, which offers one or more resources (services) for access by users on other systems in the local area network (LAN).

In addition to being able to set up your system to allow users on other systems to access its services, you can also use LATCP to set up the system to allow its users to access services on other systems in the LAN. In this case, the system can act like a terminal server: it can manage multiple user sessions simultaneously for connections to services on other nodes.

You can use LATCP to set up your system to support incoming access only, outgoing access only, or both incoming and outgoing access. You can also set up your system so that it supports neither incoming nor outgoing access.

When you set up your system to support outgoing access, the LAT software manages a database of LAT services and nodes. The software builds the database when you enable outgoing access on your node. The software begins to collect LAT **service announcements**—multicast messages sent by LAT service nodes—and builds the database based on these service announcements. You can use LATCP to display the services and nodes in this database and to control the size of the database. Allow outgoing access on systems that can tolerate the additional overhead, such as standalone systems.

Use LATCP to do the following:

- Specify operational characteristics for your node and its services
- Turn the state of the LAT port driver (LTDRIVER) on and off
- Display the status of LAT services and service nodes in the network
- Display the status of links created on your LAT node
- Display the status of your LAT node
- Show and zero LAT counters

- Create, delete, and manage LAT ports
- Recall previously entered LATCP commands so that you can execute them again without having to reenter them
- Create subprocesses so that you can execute DCL commands without exiting from LATCP

With the LAT protocol, you can set up LAT application ports on the local node so that users can access printers and other asynchronous devices that are connected to LAT terminal servers or service nodes on the LAN. The remote devices must be configured appropriately.

22.3.1 Invoking and Exiting LATCP

Enter the following command to invoke LATCP:

```
$ RUN SYS$SYSTEM:LATCP
LATCP>
```

At the LATCP> prompt, you can enter LATCP commands. To exit LATCP, enter EXIT or press Ctrl/Z at the LATCP> prompt.

You can also execute a single LATCP command by using a DCL string assignment statement, as shown in the following example:

```
$ LCP :== $LATCP
$ LCP SET NODE/STATE=ON
```

LATCP executes the SET NODE command and returns control to DCL.

22.3.2 LATCP Commands

Table 22–1 summarizes the LATCP commands.

Table 22–1 Summary of LATCP Commands

Command	Function
ATTACH	Transfers control from your current process to the specified process.
CREATE LINK	Creates LAT data links.
CREATE PORT	Creates an application port or dedicated port.
CREATE SERVICE	Creates a service on a service node.
DEFINE/KEY	Assigns a command string to a function key on your keypad.
DELETE LINK	Deletes a LAT data link from a node.
DELETE PORT	Deletes an application port or dedicated port.
DELETE SERVICE	Deletes a service on a service node.
EXIT	Returns the user to DCL command level.
HELP	Displays help text for LATCP commands.
RECALL	Recalls LATCP commands that you entered previously so that you can execute them again.
REFRESH	Refreshes your display screen, for example, after your display has been overwritten by output from some other source.
SET LINK	Modifies characteristics of LAT data links.

(continued on next page)

Table 22–1 (Cont.) Summary of LATCP Commands

Command	Function
SET NODE	Specifies LAT characteristics for a node.
SET PORT	Maps a logical port on a node to either a remote device on a terminal server or a special application service on a remote LAT service node.
SET SERVICE	Changes service characteristics.
SHOW LINK	Displays the characteristics of links on your node.
SHOW NODE	Displays the characteristics of nodes.
SHOW PORT	Displays port characteristics.
SHOW SERVICE	Displays characteristics of LAT services known to your node.
SPAWN	Creates a subprocess.
ZERO COUNTERS	Resets the node counters, service counters, and link counters maintained by your node.

For detailed information about LATCP commands and qualifiers, see the *OpenVMS System Management Utilities Reference Manual.*

22.4 Starting Up the LAT Protocol

As system manager, you start up the LAT protocol and configure your node as a service node by executing the command procedure SYS$STARTUP:LAT$STARTUP. This procedure executes the following two procedures:

1. LAT$CONFIG.COM, to load the LAT terminal driver LTDRIVER and create the LATACP process
2. LAT$SYSTARTUP.COM, to execute LATCP commands that define LAT characteristics

How to Perform This Task

To make sure the LAT protocol is started each time the system boots, add a command to execute this procedure in the general-purpose, site-specific startup command procedure, described as follows. (See Section 5.2.1 for more detailed information about this command procedure, including the file specification used to identify it in your operating system.)

To set up your node as a LAT service node and start the LAT protocol software on your system each time the system boots, edit the general-purpose, site-specific startup command procedure to add the following line:

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

When the general-purpose, site-specific startup command procedure executes this command, it invokes LAT$STARTUP.COM, which in turn invokes the LAT$CONFIG and LAT$SYSTARTUP command procedures.

You can append any of the following arguments to the command line that invokes LAT$STARTUP to specify unique LAT characteristics for your node. The procedure will pass these arguments to LAT$SYSTARTUP.COM to define the LAT characteristics you specify.

```
$ @SYS$STARTUP:LAT$STARTUP "P1" "P2" "P3" "P4" "P5"
```

Digital recommends that you modify LAT$SYSTARTUP.COM directly, rather than passing parameters in *P1* through *P5*. However, if you choose to use P1 through P5, the arguments have the following meanings:

Argument	Format	Meaning
P1	Service-name	Name of the service. For clustered service nodes, use the cluster alias as the service name. For independent service nodes, use the DECnet node name. LAT$SYSTARTUP.COM uses the argument P1 to assign a service name to the node (with the LATCP CREATE SERVICE command).
P2–P4	Any of the following:	LAT$SYSTARTUP.COM uses the arguments to assign LAT node characteristics (with the LATCP SET NODE command).
	/IDENTIFICATION= "string"	Description of the node and its services that is advertised over the local area network (LAN). The default is the string defined by the logical name SYS$ANNOUNCE. Make sure you include five sets of quotation marks around the identification string. For example: `"/IDENTIFICATION=" -` `"""""Official system center"""""`
	/GROUPS=(ENABLE=*group-list*)	Terminal server groups qualified to establish connections with the service node. By default, group 0 is enabled.
	/GROUPS=(DISABLE=*group-list*)	Removes previously enabled terminal server groups. If you are specifying the preceding qualifier to enable groups, you can combine the qualifiers into one, as shown in the example that follows this table.
P5	Any qualifiers valid with the CREATE SERVICE command	LAT$SYSTARTUP.COM uses this argument to assign service characteristics with the LATCP CREATE SERVICE command You can specify the /IDENTIFICATION, /LOG, and /STATIC_ RATING qualifiers. Specify several qualifiers as shown in the following example: `"/IDENTIFICATION=" -` `"""""Official system node""""" -` `/STATIC_RATING=250"`

Note that if you want to do any of the following LAT network tasks, you must edit LAT$SYSTARTUP.COM (described in Section 22.5):

- Set up LAT printers.
- Create special application services.
- Set up the node to allow outgoing connections (to support the SET HOST/LAT command).

For a full description of LATCP commands and qualifiers, see the *OpenVMS System Management Utilities Reference Manual*.

Example

The following command creates the service OFFICE on the service node MOE, which is part of the OFFICE cluster (refer to Figure 22–1):

```
$ @SYS$STARTUP:LAT$STARTUP OFFICE
```

22.5 Customizing LAT Characteristics

To define special LAT characteristics for your node, edit the site-specific command procedure SYS$MANAGER:LAT$SYSTARTUP.COM. This command procedure contains LATCP commands that define LAT characteristics. LAT$SYSTARTUP.COM is invoked when you execute the LAT$STARTUP command procedure. As explained in Section 22.4, you typically execute LAT$STARTUP.COM from the general-purpose, site-specific startup command procedure.

You do not need to edit LAT$SYSTARTUP.COM if you want your node to be a LAT service node that only supports incoming connections from interactive terminals. You can assign a service name and other characteristics by specifying parameters when you invoke the command procedure SYS$STARTUP:LAT$STARTUP, as described in Section 22.4.

However, you can edit LAT$SYSTARTUP.COM to add LATCP commands that customize LAT characteristics for your node, for example:

Task	For More Information
Create more than one service	Section 22.5.1
Create logical ports for special application services and printers	Section 22.5.2
Enable outgoing LAT connections to support the SET HOST/LAT command	Section 22.5.3
Tailor node characteristics[1]	Section 22.5.4

[1]For example, to assign special service announcements or LAN links (using the SET NODE and SET LINK commands).

Caution

Do not edit the command procedures LAT$STARTUP.COM and LAT$CONFIG.COM. These are procedures supplied by Digital to perform functions necessary for the LAT protocol to run correctly. Edit only LAT$SYSTARTUP.COM to define LAT characteristics specific to your site.

If you edit LAT$SYSTARTUP.COM, you should add only LATCP commands. In addition, you should conform to the order of commands in the template file SYS$MANAGER:LAT$SYSTARTUP.TEMPLATE. Section 22.5.4 provides a sample edited LAT$SYSTARTUP procedure. The *OpenVMS System Management Utilities Reference Manual* contains full descriptions of all the LATCP commands you can include in LAT$SYSTARTUP.COM.

22.5.1 Creating Additional Services

The LAT$SYSTARTUP.COM provided by Digital creates one service. This can be a primary service, one through which users can access the general computing environment. It can also be a special application service, such as a data entry program or an online news service. The procedure creates the service with the same name as that of your node, unless you specify a unique service name as an argument to the @SYS$STARTUP:LAT$STARTUP.COM command, as explained in Section 22.4.

How to Perform This Task

To create services in addition to the one provided in LAT$SYSTARTUP.COM, use the CREATE SERVICE commands, which you can add to LAT$SYSTARTUP.COM. Note that if you create an application service, Digital recommends that you assign the name of the application program. For more information on the LATCP command CREATE SERVICE, see the *OpenVMS System Management Utilities Reference Manual.*

Example

```
$ LCP :== $LATCP
$ LCP CREATE SERVICE NEWS/IDENT/APPLICATION
```

22.5.2 Setting Up Ports

The LAT$SYSTARTUP.COM file provided by Digital includes sample commands to create logical ports on the service node and associates them with physical ports or services on the terminal server node. These ports can be used for application services and remote printers.

How to Perform This Task

To create ports, enable the sample commands by removing the exclamation points (!) that precede them in the LAT$SYSTARTUP.COM file, or add similar CREATE PORT and SET PORT commands to that file to meet your needs. For information on the LATCP commands CREATE PORT and SET PORT, see the *OpenVMS System Management Utilities Reference Manual.*

Note

Digital strongly recommends that you create application and dedicated ports *after* the LATCP command SET NODE/STATE=ON is executed. This minimizes nonpaged pool memory usage and eliminates the possibility of creating duplicate ports.

Note that you may encounter the following error when attempting to create an application port (with a command such as LCP CREATE PORT LTA5001: /APPLICATION, for example):

```
%LAT-W-CMDERROR, error reported by command executor
-SYSTEM-F-DUPLNAM, duplicate name
```

This error indicates that the LAT application port you are trying to create is already created by some other application. This application could be LATCP itself (LATCP's port, LATCP$MGMT_PORT, is used to communicate with LTDRIVER).

To avoid this error, make sure the SET NODE/STATE=ON command is executed before any commands that create application or dedicated ports. You can also use the LATCP command SET NODE/DEVICE_SEED. For more information on the

SET NODE/DEVICE_SEED command, see the *OpenVMS System Management Utilities Reference Manual*.

22.5.2.1 Setting Up Printers

If you set up a port for a printer, you must also perform the following tasks:

1. Create a spooled output queue for the printer.
2. Add a command to start the queue to the startup command procedure that starts your queues, or to the general-purpose, site-specific startup command procedure.

These tasks are described in Chapter 13.

22.5.2.2 Setting Up Special Application Services

To establish a special application service, include the /DEDICATED qualifier when defining a LAT port. The application program to which the service connects must define the same dedicated port. For example, the following commands set up ports for an application service called NEWS:

```
$ LCP :== $LATCP
$ LCP CREATE PORT LTA333:/DEDICATED
$ LCP SET PORT LTA333:/SERVICE=NEWS
```

Before application services can be available to user terminals on the LAT network, you must start the application program. You usually add commands to SYLOGIN.COM to do this.

22.5.3 Enabling Outgoing LAT Connections

By default, outgoing LAT connections are disabled on a node. If you want to allow users to use the SET HOST/LAT connection to establish LAT connections from the node, you must edit LAT$SYSTARTUP.COM to enable outgoing connections. For more details on using the SET HOST/LAT command for outgoing LAT connections, see the description of that command in the *OpenVMS DCL Dictionary*.

Commands to enable outgoing connections are included in the LAT$SYSTARTUP.COM file provided by Digital. Enable the command of your choice by removing the exclamation point (!) that precedes it, or add a similar command to meet your needs. For more information, see the /CONNECTIONS and /USER_GROUPS qualifiers to the LATCP command SET NODE in the *OpenVMS System Management Utilities Reference Manual*.

To attain optimal SET HOST/LAT performance and forward port performance, you should set the system parameter TTY_ALTYPAHD to 1500 and reboot.

If you want to set up your node only as a service node with incoming connections enabled, you do not need to edit LAT$SYSTARTUP.COM. However, you might edit LAT$SYSTARTUP.COM to do one or more of the following tasks:

- Create more than one service on a node
- Create special application services
- Set up LAT printers
- Enable outgoing LAT connections (to allow a node to act as a terminal server node)
- Tailor node characteristics; for example, to assign special service announcements or connections to the LAN

22.5.4 Sample Edited LAT$SYSTARTUP.COM Procedure

The following is a sample of an edited LAT$SYSTARTUP.COM procedure that creates services, creates and sets ports, and sets nodes to allow incoming and outgoing connections.

```
$! Copyright (c) 1992 Digital Equipment Corporation.  All rights reserved.
$!
$!    LAT$SYSTARTUP.COM -- LAT Startup Commands Specific to Site
$!
$!    Use this command procedure to customize the LAT characteristics for
$!    the local node.  These commands, which should serve as examples,
$!    will set up a LAT service name SYS$NODE and default identification
$!    SYS$ANNOUNCE.  The LAT service name and identification will default
$!    to SYS$NODE and SYS$ANNOUNCE unless you specify a service name and
$!    identification as arguments to the command line that invokes
$!    LAT$STARTUP.COM:
$!                    $ @SYS$STARTUP:LAT$STARTUP
$!
$!    You can specify other node and service characteristics (such as group
$!    codes) as arguments to this command line, as shown below.
$!
$!        Argument     Function
$!        ---------    --------
$!
$!           P1        Name of the service to be created.  If not supplied, a
$!                     service will be created with the same name as the node.
$!
$!        P2,P3,P4     Parameters and qualifiers to the SET NODE command.
$!
$!           P5        Parameters and qualifiers to the SET SERVICE command.
$!                     P5 is only used if P1 is specified.  More than one
$!                     argument may be supplied by enclosing the string in
$!                     quotes.
$!
$!    Example: $ @SYS$STARTUP:LAT$STARTUP HAWK "/IDENTIFICATION=" -
$!   """""Development node"""""
$!
$!    Please review and edit this file for possible additions and deletions
$!    that you wish to make.  Future software updates will not overwrite the
$!    changes made to this file.
$!
$ required_privileges = "OPER"
$ prev_privs = f$setprv(required_privileges)
$ if .not. f$privilege(required_privileges) then goto no_privileges
$ lcp := $latcp
$!
$! ---------------------  Modify Node Characteristics  ------------------------
$!
$ lcp set node 'p2' 'p3' 'p4'
$!
$! Some examples:
$!
$! ** Allow incoming connections only
$!
$! lcp set node /connections=incoming /groups=(enable=(12,40,43,73),disable=0)
$! lcp set node /connections=incoming /groups=enable=(0-255)
$!
$ LCP SET NODE /CONNECTIONS=INCOMING /GROUPS=(ENABLE=(12,40,43,73),DISABLE=0)
$!
$! ** Allow outgoing connections only
$!
$! lcp set node /connections=outgoing /user_groups=enable=(24,121-127)
$! lcp set node /connections=outgoing /user_groups=(enable=0-255) /node_limit=50
$!
```

```
$! ** Enable incoming and outgoing connections
$!
$! lcp set node /connections=both /group=enable=(43,73) /user=enable=(44,56)
$! lcp set node /connections=both /group=enable=(0-255) /user=enable=(0-255)
$!
$!
$! --------------------  Modify Service Characteristics  ----------------------
$!
$ if p1 .eqs. ""
$ then
$    lcp create service
$ else
$     lcp create service 'p1' 'p5'
$ endif
$! -------------------------  Start LAT Protocol  -----------------------------
$!
$ lcp set node /state=on
$!
$!
$! -------------------------  Create and Map Ports  ---------------------------
$!
$! Some examples:
$!
$! lcp create port lta101: /dedicated
$! lcp create port lta102: /application
$! lcp create port lta103: /application
$! lcp create port /nolog/logical=(name=ln03$mgmt, table=system, mode=executive)
$
$ LCP CREATE PORT LTA1: /NOLOG
$ LCP CREATE PORT LTA20: /NOLOG
$
$! lcp set port lta101: /dedicated /service=graphics
$! lcp set port lta102: /node=server_1 /port=port_1
$! lcp set port lta103: /node=server_2 /service=laser
$! lcp set port ln03$mgmt: /node=server_3 /service=ln03_printers
$!
$ LCP SET PORT LTA1: /APPLICATION/NODE=TERM_SERVER_1 /PORT=PORT_6
$ LCP SET PORT LTA20: /APPLICATION/NODE=TERM_SERVER_2 /PORT=PORT_6
$!
$exit:
$ prev_privs = f$setprv(prev_privs)
$ exit
$!
$no_privileges:
$ write sys$output "Insufficient privileges to execute LATCP commands."
$ write sys$output "Requires ",required_privileges," privileges."
$ goto exit
```

22.6 Managing the LATACP Database Size

On OpenVMS nodes, another component of the LAT software, the LAT Ancillary Control Process (LATACP), maintains the database of available nodes and services. The nodes and services can be those that are multicast from remote LAT nodes, or they can consist of the local node and one or more local services that you create on your own system. The maximum size of this database is dependent on the value of the system parameter CTLPAGES.

After you enter a LATCP command, you might get the following response:

```
%LAT-W-CMDERROR, error reported by command executor
-LAT-F-ACPNOCTL, insufficient resources - ACP CTL/P1 space limit reached
```

If so, this signifies that the database size has reached the CTLPAGES limit. You can correct the situation in one of three ways:

- Reduce the size of the database by reducing the node limit. Use the LATCP command SHOW NODE to display the node limit; use the command SET NODE/NODE_LIMIT to change it. For more information, see the *OpenVMS System Management Utilities Reference Manual*.
- Reduce the size of the database by reducing the user group codes that are enabled on the node. Use the LATCP command SHOW NODE to display the enabled user group codes; use the command SET NODE/USER_GROUPS=DISABLE to disable some of them. For more information, see the *OpenVMS System Management Utilities Reference Manual*.

  If you choose this step, you must also edit your startup procedures to change the user groups that are enabled each time the system reboots. For more information, see Section 22.5.
- Extend the size of the database by increasing the value of the system parameter CTLPAGES. As a general rule, note that every unit of CTLPAGES that you increase is roughly equivalent to six additional nodes or services that will be stored in the database.

  After you change CTLPAGES, you must reboot the system for the changed value to take effect. Make sure you add the increased value of CTLPAGES to the AUTOGEN parameter file MODPARAMS.DAT. For more information on changing values of system parameters, see Section 14.2.

23 Managing DECdtm Services

This chapter describes what you must do if you want to run software that uses DECdtm services. Software products that can currently use DECdtm services include ACMS, DBMS, DECintact, Rdb, and RMS Journaling.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Planning transaction logs	Section 23.2
Creating transaction logs	Section 23.3
Monitoring transaction performance	Section 23.4
Checking whether a transaction log is too small	Section 23.5
Changing the size of a transaction log	Section 23.6
Moving a transaction log	Section 23.7
Creating a transaction log for a new node	Section 23.8
Removing a node	Section 23.9
Disabling DECdtm services	Section 23.10
Enabling DECdtm services	Section 23.11

The map in Figure 23–1 shows which of these tasks you need to do, and the order to do them in.

This chapter explains the following concept:

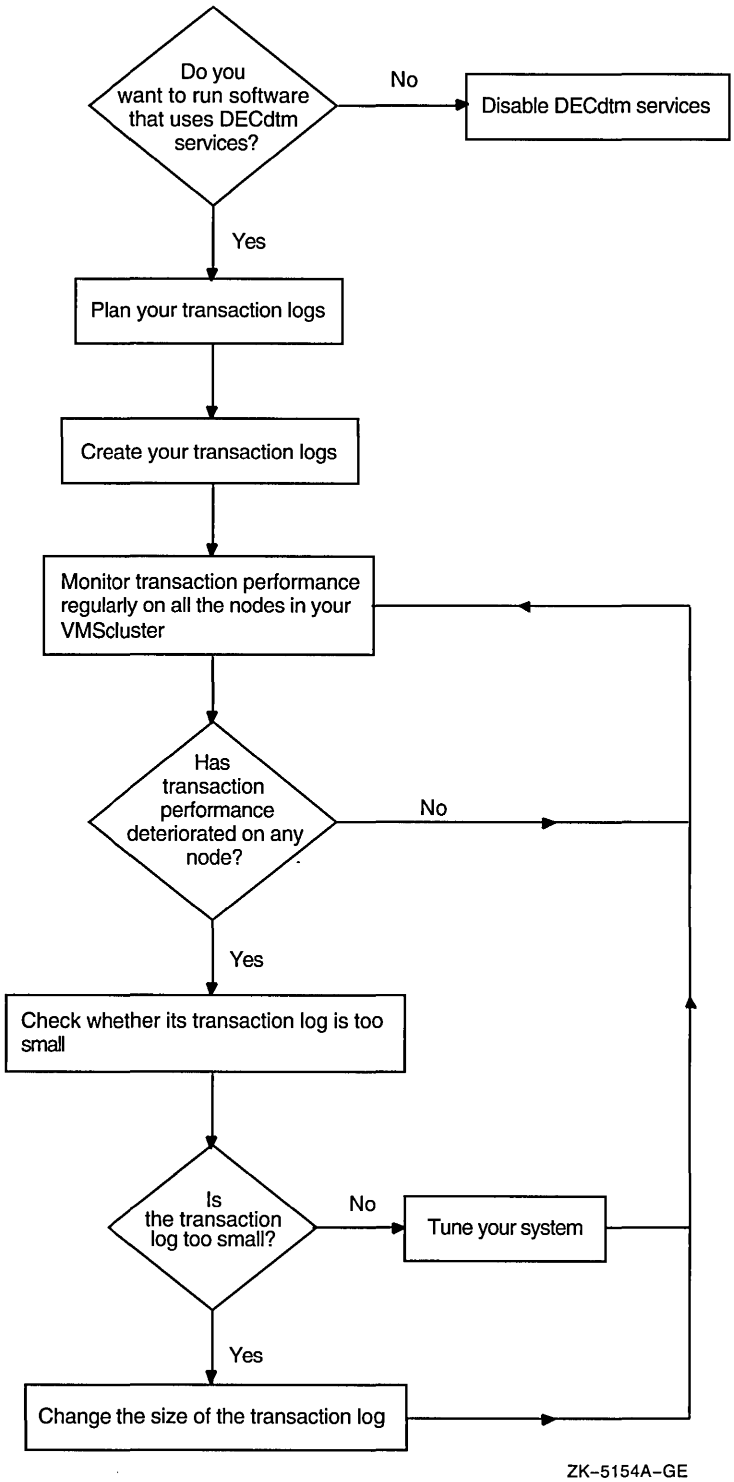
Concept	Section
Understanding transaction logs	Section 23.1

23.1 Understanding Transaction Logs

Before any software can use DECdtm services, you need to create a transaction log for each node in your VMScluster environment. A **transaction log** is a file that stores information about the transactions performed on a particular node.

The system uses the logical name SYS$JOURNAL to determine where the transaction logs are. You must define SYS$JOURNAL as a search list of all the directories containing transaction logs, and keep this search list up to date.

Figure 23–1 Managing DECdtm Services



23.2 Planning Transaction Logs

The size and location of your transaction logs can affect transaction performance. Before you create the transaction logs, plan how big to make them and where you want to put them.

Later, you can change the size of a transaction log, or move it. However, to do either of these, you need to reboot the node (see Section 23.6 and Section 23.7). Careful planning at this stage reduces the need for later changes.

This section describes how to plan transaction logs:

Task	For More Information
Deciding how big to make your transaction logs	Section 23.2.1
Deciding where to put your transaction logs	Section 23.2.2

23.2.1 Deciding How Big to Make the Transaction Logs

Note

These guidelines give only an approximate size. When planning transaction logs, Digital recommends that you overestimate, rather than underestimate, the size of the transaction log.

When you create a transaction log, you can specify its size in blocks. The default size is 4000 blocks; this gives acceptable performance on most systems.

If you know the expected rate of transactions, Digital suggests the following algorithm to calculate the transaction log size:

$$size = 40 * rate$$

where:

size is the size of the transaction log in blocks.

rate is the average number of transactions executed per second.

If you do not know the rate of transactions, accept the default size of 4000 blocks.

Example

In this example, the average number of transactions executed per second is 120 transactions. The recommended size for the transaction log is 4800 blocks, calculated as follows:

$$size = 40 * 120 = 4800$$

23.2.2 Deciding Where to Put the Transaction Logs

If possible, distribute the transaction logs across different disks. Having more than one transaction log on a disk can lead to poor transaction performance.

For each transaction log, if possible choose a disk that is:

Attribute	Description
Fast	Use a high-performance disk, such as a solid-state disk, that is not heavily used.
Highly available	You achieve high availability by having multiple access paths to the data. Use a disk that can be accessed by the other nodes in your VMScluster environment when the node that the transaction log belongs to is down. This reduces the time that transactions running on other nodes in the VMScluster environment are blocked while waiting for the node to recover.
Reliable	You achieve reliability by keeping multiple copies of the data. A shadowed disk is more reliable than a nonshadowed disk, but may be slower because transaction logs are almost exclusively write-only.

You may need to choose between speed and either availability or reliability. For example, if the node is a workstation, you may choose to sacrifice speed for availability and reliability by putting the node's transaction log on a shadowed HSC-based disk, instead of on a faster disk attached to the workstation.

Note

Make sure that the disk has sufficient contiguous space to hold the transaction log. A discontiguous transaction log leads to poor transaction performance.

23.3 Creating Transaction Logs

Before any software can use DECdtm services, you must create a transaction log for each node in your VMScluster environment.

You must also create a transaction log for each new node you add to your VMScluster. For instructions on how to create a transaction log for a new node, see Section 23.8.

Caution

Removing a node from the VMScluster after you have created the transaction logs can lead to data corruption. For instructions on how to remove a node safely, see Section 23.9.

How to Perform This Task

1. Using the guidelines given in Section 23.2, decide how big to make the transaction logs and where to put them.
2. Log in to any node in the VMScluster.
3. Enable the SYSPRV and OPER privileges.

4. Make sure that the disks on which you want to create the transaction logs are mounted clusterwide.
5. Decide which directories you want to create the transaction logs in. You may want to create new directories specifically for the transaction logs.
6. Define SYS$JOURNAL to point to the directories in which you want to create the transaction logs. Use SYSMAN to do this for each node in the VMScluster environment. Set your SYSMAN environment and profile, then enter a DO command in the following format:

   DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL dirspec[,...]

   where *dirspec* is the full specification of a directory in which you want to create one or more transaction logs. You must list all the directories that will contain transaction logs. You can list them in any order.

   For example:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
_SYSMAN> DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES]
SYSMAN> EXIT
```

7. Edit the SYS$STARTUP:SYLOGICALS.COM command procedure to include the SYS$JOURNAL definition.

   If you have created node-specific versions of SYLOGICALS.COM, edit all the versions.
8. Create one transaction log for each node in your VMScluster, using LMCP's CREATE LOG command in the following format:

   CREATE LOG [/SIZE=size] dirspecSYSTEM$node.LM$JOURNAL

   where:

size	is the size of the transaction log in blocks. By default, the size of the transaction log is 4000 blocks.
dirspec	is the full specification of the directory in which you want to create the transaction log.
node	is the name of the node.

   For example:

```
$ RUN SYS$SYSTEM:LMCP
LMCP> CREATE LOG/SIZE=5000 DISK$LOG1:[LOGFILES]SYSTEM$ORANGE.LM$JOURNAL
LMCP> EXIT
```

9. If you have previously disabled DECdtm services, you must enable them. See Section 23.11 for information on how to enable DECdtm services.

Example

In this example, the VMScluster has two nodes, ORANGE and RED. Assume that neither node has a node-specific version of SYLOGICALS.COM.

Decide where to put the transaction logs for nodes ORANGE and RED, and how big to make them. For example:

Node	Size of Log (in Blocks)	Directory to Contain Log
ORANGE	5000	DISK$LOG1:[LOGFILES]
RED	4000	DISK$LOG2:[LOGFILES]

Log in to either node ORANGE or node RED and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(SYSPRV,OPER)
$ MOUNT/CLUSTER DUA1: LOG1
$ MOUNT/CLUSTER DUA2: LOG2
$ CREATE/DIRECTORY DISK$LOG1:[LOGFILES]
$ CREATE/DIRECTORY DISK$LOG2:[LOGFILES]
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
_SYSMAN> DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES]
SYSMAN> EXIT
```

Edit the SYS$STARTUP:SYLOGICALS.COM command procedure to include the following lines:

```
$ !
$ DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES]
$ !
```

Then enter the following commands:

```
$ RUN SYS$SYSTEM:LMCP
LMCP> CREATE LOG/SIZE=5000 DISK$LOG1:[LOGFILES]SYSTEM$ORANGE.LM$JOURNAL
LMCP> CREATE LOG DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$JOURNAL
LMCP> EXIT
```

23.4 Monitoring Transaction Performance

Changes to your system can affect transaction performance. Once a month, monitor transactions on all the nodes in your VMScluster environment to make sure that transaction performance has not deteriorated.

How to Perform This Task

1. Monitor transactions, using the Monitor utility's MONITOR TRANSACTION command in the following format:

   MONITOR TRANSACTION/SUMMARY[=summary-file]/ENDING=end-time/NODE=node[,...]

   where:

summary-file	is the file specification of the summary file. Information about transactions is summarized and recorded in the summary file. If you omit the file specification, the information is recorded in MONITOR.SUM in your default directory.
end-time	is the time that the monitoring session ends.
node	is the name of a node. List all the nodes in your VMScluster.

   For the best results, monitor transactions for 24 hours at a time.

You can monitor transactions in batch mode by including the MONITOR TRANSACTION command in a command procedure.

2. Examine the information recorded in the summary file.

   The summary file contains values for a number of different data items. Note the following rates for each node:

   - The average end rate.

     This is the average number of transactions completed per second.

   - The average completion rates.

     These are the average numbers of transactions completed in the following times:

     – Less than 1 second
     – Between 1 and 2 seconds
     – Between 2 and 3 seconds
     – Between 3 and 4 seconds
     – Between 4 and 5 seconds
     – More than 5 seconds

   Keep a record of these values.

   For a full description of the data items, see the *OpenVMS System Management Utilities Reference Manual*.

3. Compare the results from this monitoring session with the results from previous sessions. For the same work load, the rate and duration of transactions should remain about the same. Indications of performance degradation are:

   - The average end rate decreases.
   - The average duration increases.

     To find out if the average duration of transactions has increased, compare the average completion rates. If a greater proportion of the transactions take longer to complete, the average duration of transactions has increased.

   Note any trends over a number of monitoring sessions. Variations from one monitoring session to the next are probably due to variations in the work load on your system.

   If you suspect that transaction performance has deteriorated on any node, check whether its transaction log is too small (see Section 23.5).

   If the transaction log is big enough, but transaction performance still deteriorates, you may need to tune your system. See the *Guide to OpenVMS Performance Management* for information on tuning your system.

Example

In this example, the VMScluster has two nodes, ORANGE and RED. Enter the following command:

```
$ MONITOR TRANSACTION/SUMMARY=DISK$LOG1:[LOGFILES]TRANSACTIONS.SUM -
_$ /ENDING="+24"/NODE=(ORANGE,RED)
```

Examine the contents of the summary file. The following shows an example summary file:

```
                   DISTRIBUTED TRANSACTION STATISTICS
                             on node ORANGE      From: 16-AUG-1994 14:23:51
                                SUMMARY          To:   17-AUG-1994 14:23:51

                                   CUR        AVE        MIN        MAX

Start Rate                        49.02      43.21      31.30      49.02
Prepare Rate                      48.70      43.23      30.67      48.70
One Phase Commit Rate              0.00       0.00       0.00       0.00
Total Commit Rate                 48.70      43.19      31.30      48.70
Abort Rate                         0.00       0.00       0.00       0.00
End Rate                          48.70      43.19      31.30      48.70
Remote Start Rate                  0.00       0.00       0.00       0.00
Remote Add Rate                    0.00       0.00       0.00       0.00

Completion Rate   0-1             21.42      13.57       0.63      21.42
 by Duration      1-2             25.97      29.15      24.59      33.87
 in Seconds       2-3              1.29       0.47       0.00       4.47
                  3-4              0.00       0.00       0.00       0.00
                  4-5              0.00       0.00       0.00       0.00
                   5+              0.00       0.00       0.00       0.00

                              SUMMARIZING

                   DISTRIBUTED TRANSACTION STATISTICS
                             on node RED         From: 16-AUG-1994 14:23:52
                                SUMMARY          To:   17-AUG-1994 14:23:52

   .
   .
   .
```

Note the following values:

- The average end rate.

  For node ORANGE, the average end rate is 43.19 transactions per second.

- The average completion rates.

  For node ORANGE, the average completion rates are as follows:

  13.57 transactions completed in 0 to 1 seconds
  29.15 transactions completed in 1 to 2 seconds
  0.47 transactions completed in 2 to 3 seconds

Compare the results from this monitoring session to those of previous sessions:

Session	End Rate	Completion Rates		
		0–1 Secs	1–2 Secs	2–3 Secs
June	42.13	12.98	28.13	1.02
July	38.16	10.35	25.80	2.01
August	43.19	13.57	29.15	0.47

The results for node ORANGE show no signs of deteriorating performance.

23.5 Checking Whether a Transaction Log Is Too Small

If transaction performance degrades on any node, check to see if its transaction log is too small.

How to Perform This Task

1. Log in to the node that the transaction log belongs to.
2. Enable the CMKRNL privilege.
3. Check how many times the transaction log has stalled, using the LMCP utility's SHOW LOG command:

   ```
   $ RUN SYS$SYSTEM:LMCP
   LMCP> SHOW LOG/CURRENT
   ```

   Note the number of checkpoints and stalls that have occurred, then wait for 5 minutes before repeating the SHOW LOG/CURRENT command. Again, note the number of checkpoints and stalls.

   If the number of checkpoints is the same in both readings, wait until the system is busier, then try this task again.

   If the number of checkpoints has increased, and the number of stalls has increased by more than 1, the transaction log is too small.
4. If the transaction log is too small, increase its size. For information on how to change the size of a transaction log, see Section 23.6.
5. Check the transaction log again after you have changed its size. If it continues to stall more than once every 5 minutes, it is still too small.

Example

This example shows how to check whether node ORANGE's transaction log is too small.

Log in to node ORANGE and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=CMKRNL
$ RUN SYS$SYSTEM:LMCP
LMCP> SHOW LOG/CURRENT

Checkpoint starts/ends           2464/2464
Stall starts/ends                  21/21
Log status: no checkpoint in progress, no stall in progress.
```

The number of checkpoints is 2464, and the transaction log has stalled 21 times. Wait for 5 minutes, then repeat the SHOW LOG/CURRENT command:

```
LMCP> SHOW LOG/CURRENT

Checkpoint starts/ends           2514/2514
Stall starts/ends                  28/28
Log status: no checkpoint in progress, no stall in progress.
```

The number of checkpoints has increased since the previous reading, and the transaction log has now stalled 28 times, an increase of 7. This means that the transaction log is too small.

23.6 Changing the Size of a Transaction Log

If a transaction log is too small, you need to increase its size.

How to Perform This Task

Caution

Follow all the steps carefully. Taking shortcuts can lead to data corruption.

1. Log in to the node that the transaction log belongs to.
2. If you have the SETPRV privilege, enable the SYSPRV privilege.

   If you do not have the SETPRV privilege, enable the following privileges: CMKRNL, EXQUOTA, LOG_IO, OPER, SYSNAM, SYSPRV, TMPMBX, and WORLD.
3. Rename the transaction log:

   RENAME dirspecSYSTEM$node.LM$JOURNAL dirspecSYSTEM$node.LM$OLD

   where:

dirspec	is the full specification of the directory containing the transaction log.
node	is the name of the node that the transaction log belongs to.

   For example:

   ```
   $ RENAME DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$JOURNAL -
   _$ DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$OLD
   ```
4. Shut down the node using the following command:

   ```
   $ @SYS$SYSTEM:SHUTDOWN.COM
   ```

   The shutdown procedure prompts you with a series of questions. If you are in a VMScluster environment, specify that an automatic reboot is not performed. If you have a single-node system, specify that an automatic reboot is performed.

   For instructions on how to answer the questions, see Section 4.8.1.
5. If you are in a VMScluster environment, log in to another node. If you have a single-node system, log in to the same node.
6. Enable the CMKRNL and SYSPRV privileges.
7. Change the size of the transaction log, using the LMCP utility's CONVERT LOG command in the following format:

   CONVERT LOG/SIZE=size dirspecSYSTEM$node.LM$OLD dirspecSYSTEM$node.LM$JOURNAL

   where:

size	is the new size of the transaction log in blocks.
dirspec	is the full specification of the directory containing the transaction log.
node	is the name of the node that the transaction log belongs to.

For example:

```
$ RUN SYS$SYSTEM:LMCP
LMCP> CONVERT LOG/SIZE=6000 DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$OLD -
_LMCP> DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$JOURNAL
LMCP> EXIT
```

8. If you are in a VMScluster environment, perform a nonstop boot of the node that the transaction log belongs to.
9. Archive the old transaction log, SYSTEM$*node*.LM$OLD.

Example

In this example, node RED is in a VMScluster. Its transaction log is in DISK$LOG2:[LOGFILES]. The following instructions show how to change the size of node RED's transaction log to 6000 blocks. Assume that you do not have the SETPRV privilege.

Log in to node RED and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(CMKRNL,EXQUOTA,LOG_IO,OPER,SYSNAM,SYSPRV,TMPMBX,WORLD)
$ RENAME DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$JOURNAL -
_$ DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$OLD
$ @SYS$SYSTEM:SHUTDOWN.COM
   .
   .
   .
Should an automatic system reboot be performed [NO]? NO
```

Now log in to another node in the VMScluster and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(CMKRNL,SYSPRV)
$ RUN SYS$SYSTEM:LMCP
LMCP> CONVERT LOG/SIZE=6000 DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$OLD -
_LMCP> DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$JOURNAL
LMCP> EXIT
```

Next, perform a nonstop boot of node RED.

Finally, archive DISK$LOG2:[LOGFILES]SYSTEM$RED.LM$OLD, the old transaction log.

23.7 Moving a Transaction Log

You may want to move a transaction log if:

- You want to place the transaction log on a faster disk
- You want to redistribute the work load on your disks

How to Perform This Task

Caution

Follow all the steps carefully. Taking shortcuts can lead to data corruption.

1. Using the guidelines given in Section 23.2.2, decide where you want to move the transaction log to.
2. Log in to the node that the transaction log belongs to.
3. If you have the SETPRV privilege, enable the SYSPRV privilege.

If you do not have the SETPRV privilege, enable the following privileges: CMKRNL, EXQUOTA, LOG_IO, OPER, SYSNAM, SYSPRV, TMPMBX, and WORLD.

4. Make sure that the disk you want to move the transaction log to is mounted clusterwide.
5. Decide which directory you want to move the transaction log to. You may want to create a new directory specifically for the transaction log.
6. Rename the transaction log:

   RENAME dirspecSYSTEM$node.LM$JOURNAL dirspecSYSTEM$node.LM$OLD

   where:

dirspec	is the full specification of the directory containing the transaction log.
node	is the name of the node that the transaction log belongs to.

   For example:

   ```
   $ RENAME DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$JOURNAL -
   _$ DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$OLD
   ```

7. Shut down the node using the following command:

   ```
   $ @SYS$SYSTEM:SHUTDOWN.COM
   ```

   The shutdown procedure prompts you with a series of questions. If you are in a VMScluster environment, specify that an automatic reboot is not performed. If you have a single-node system, specify that an automatic reboot is performed.

   For instructions on how to answer the questions, see Section 4.8.1.

8. If you are in a VMScluster environment, log in to another node. If you have a single-node system, log in to the same node.
9. Enable the CMKRNL, SYSPRV, and OPER privileges.
10. Make sure that SYS$JOURNAL points to the directory that you want to move the log to. If it does not, redefine SYS$JOURNAL for each node in the VMScluster environment. Use SYSMAN to do this for each node in the VMScluster. Set your SYSMAN environment and profile, then enter a DO command in the following format:

    DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL dirspec[,...]

    where *dirspec* is the full specification of a directory containing one or more transaction logs. You must list all the directories that will contain transaction logs after you have moved the transaction log. You can list them in any order.

    For example:

    ```
    $ RUN SYS$SYSTEM:SYSMAN
    SYSMAN> SET ENVIRONMENT/CLUSTER
    SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
    SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
    _SYSMAN> DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES]
    SYSMAN> EXIT
    ```

11. If you redefined SYS$JOURNAL in step 10, edit the SYS$STARTUP:SYLOGICALS.COM command procedure to update the definition of SYS$JOURNAL.

If you have created node-specific versions of SYLOGICALS.COM, edit all the versions.

12. Move the transaction log, using LMCP's CONVERT LOG command in the following format:

    CONVERT LOG old-dirspecSYSTEM$node.LM$OLD new-dirspecSYSTEM$node.LM$JOURNAL

    where:

old-dirspec	is the full specification of the directory that currently contains the transaction log.
node	is the name of the node that the transaction log belongs to.
new-dirspec	is the full specification of the directory that you are moving the transaction log to.

    For example:

    ```
    $ RUN SYS$SYSTEM:LMCP
    LMCP> CONVERT LOG DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$OLD -
    _LMCP> DISK$LOG1:[LOGFILES]SYSTEM$BLUE.LM$JOURNAL
    LMCP> EXIT
    ```

13. If you are in a VMScluster environment, perform a nonstop boot of the node that the transaction log belongs to.
14. Archive the old transaction log, SYSTEM$*node*.LM$OLD.

Example

In this example, the VMScluster members and the locations of their transaction logs are as follows:

Node	Directory Containing Log
BLUE	DISK$LOG3:[LOGFILES]
RED	DISK$LOG2:[LOGFILES]

The following instructions show how to move node BLUE's transaction log. Assume that you do not have the SETPRV privilege, and that neither node has a node-specific version of SYLOGICALS.COM.

Decide where you want to move BLUE's transaction log to. In this example, assume that you want to move it to DISK$LOG1:[LOGFILES].

First, log in to node BLUE and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(CMKRNL,EXQUOTA,LOG_IO,OPER,SYSNAM,SYSPRV,TMPMBX,WORLD)
$ MOUNT/CLUSTER DUA1: LOG1
$ CREATE/DIRECTORY DISK$LOG1:[LOGFILES]
$ RENAME DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$JOURNAL -
_$ DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$OLD
$ @SYS$SYSTEM:SHUTDOWN.COM
    .
    .
    .
Should an automatic system reboot be performed [NO]? NO
```

Next, log in to node RED and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(CMKRNL,SYSPRV,OPER)
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
_SYSMAN> DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES]
SYSMAN> EXIT
```

Edit the SYS$STARTUP:SYLOGICALS.COM command procedure to update the SYS$JOURNAL definition. Then enter the following commands:

```
$ RUN SYS$SYSTEM:LMCP

LMCP> CONVERT LOG DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$OLD -
_LMCP> DISK$LOG1:[LOGFILES]SYSTEM$BLUE.LM$JOURNAL

LMCP> EXIT
```

Perform a nonstop boot of node BLUE.

Finally, archive DISK$LOG3:[LOGFILES]SYSTEM$BLUE.LM$OLD, the old transaction log.

23.8 Creating a Transaction Log for a New Node

For every node you add to a VMScluster that uses DECdtm services, you need to create a new transaction log.

How to Perform This Task

Before you perform this task, the new node must be configured into the VMScluster. For instructions on how to configure a node into a VMScluster, see *VMScluster Systems for OpenVMS*.

1. Using the guidelines given in Section 23.2, decide how big to make the new node's transaction log and where to put it.
2. Log in to any node in the VMScluster.
3. Enable the OPER and SYSPRV privileges.
4. Make sure that the disk on which you want to create the transaction log is mounted clusterwide.
5. Decide which directory you want to create the new transaction log in. You may want to create a new directory specifically for the transaction log.
6. Make sure that SYS$JOURNAL points to the directory that you want to create the new node's transaction log in. If it does not, redefine SYS$JOURNAL for each node in the VMScluster. Use SYSMAN to do this for each node in the VMScluster environment. Set your SYSMAN environment and profile, then enter a DO command in the following format:

   DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL dirspec[,...]

   where *dirspec* is the full specification of a directory containing one or more transaction logs. List all the directories that contain transaction logs, including the directory in which you want to create the new node's transaction log. You can list them in any order.

For example:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
_SYSMAN> DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES], DISK$LOG3:[LOGFILES]
SYSMAN> EXIT
```

7. If you redefined SYS$JOURNAL in step 6, edit the SYS$STARTUP:SYLOGICALS.COM command procedure to update the SYS$JOURNAL definition.

   If you have created node-specific versions of SYLOGICALS.COM, edit all the versions.

8. Create the transaction log, using LMCP's CREATE LOG command in the following format:

   CREATE LOG [/SIZE=size] dirspecSYSTEM$node.LM$JOURNAL

   where:

size	is the size of the transaction log in blocks. By default, the size of the transaction log is 4000 blocks.
dirspec	is the full specification of the directory in which you want to create the transaction log.
node	is the name of the new node.

   For example:

```
$ RUN SYS$SYSTEM:LMCP
LMCP> CREATE LOG/SIZE=5000 DISK$LOG3:[LOGFILES]SYSTEM$WHITE.LM$JOURNAL
LMCP> EXIT
```

Example

In this example, the VMScluster members and the locations of their transaction logs are as follows:

Node	Directory Containing Log
ORANGE	DISK$LOG1:[LOGFILES]
RED	DISK$LOG2:[LOGFILES]

WHITE is a new node. This example shows how to create a transaction log for node WHITE. Assume that none of the nodes has a node-specific version of SYLOGICALS.COM.

First, decide where you want to put WHITE's transaction log and how big to make it. For example:

Node	Size of Log (in Blocks)	Directory to Contain Log
WHITE	5000	DISK$LOG3:[LOGFILES]

Next, log in to node ORANGE, RED, or WHITE and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(OPER,SYSPRV)
$ MOUNT/CLUSTER DUA3: LOG3
$ CREATE/DIRECTORY DISK$LOG3:[LOGFILES]
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
_SYSMAN> DISK$LOG1:[LOGFILES], DISK$LOG2:[LOGFILES], DISK$LOG3:[LOGFILES]
SYSMAN> EXIT
```

Finally, edit the SYS$STARTUP:SYLOGICALS command procedure to update the SYS$JOURNAL definition. Then enter the following commands:

```
$ RUN SYS$SYSTEM:LMCP
LMCP> CREATE LOG/SIZE=5000 DISK$LOG3:[LOGFILES]SYSTEM$WHITE.LM$JOURNAL
LMCP> EXIT
```

23.9 Removing a Node

This section describes how to remove a node if you are using DECdtm services.

Caution

Follow all the steps carefully. Taking shortcuts can lead to data corruption.

How to Perform This Task

If you have a single-node system, perform steps 1 to 8 only.

In this section, the node being removed is referred to as OLDNOD.

1. Log in to node OLDNOD.
2. If you have the SETPRV privilege, enable the OPER and SYSPRV privileges.

   If you do not have the SETPRV privilege, enable the following privileges: CMKRNL, EXQUOTA, LOG_IO, OPER, SYSNAM, SYSPRV, TMPMBX, and WORLD.
3. Stop all software on OLDNOD that uses DECdtm services. To find out how to stop the software, see the documentation for those software products.
4. Determine if OLDNOD's transaction log contains any active transactions, using LMCP's DUMP command:

   ```
   $ RUN SYS$SYSTEM:LMCP
   LMCP> DUMP/ACTIVE SYSTEM$OLDNOD.LM$JOURNAL
   ```

   This command displays details of all the active transactions. The last line gives the total number of active transactions. If the transaction log does contain any active transactions:

   a. Run recovery procedures for any software on the network that uses DECdtm services. To find out how to do this, see the documentation for those software products.

   b. Check the transaction log again for active transactions, using LMCP's DUMP/ACTIVE command.

   Do not continue to the next step if the transaction log still contains active transactions.

5. Redefine SYS$JOURNAL to exclude the directory that contained OLDNOD's transaction log, unless that directory contains other transaction logs. Use SYSMAN to redefine SYS$JOURNAL for each node in the VMScluster environment. Set your SYSMAN environment and profile, then enter a DO command in the following format:

   DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL dirspec[,...]

   where *dirspec* is the full specification of a directory containing one or more transaction logs. List all the directories that contain any transaction logs other than OLDNOD's. You can list them in any order.

   For example:

   ```
   $ RUN SYS$SYSTEM:SYSMAN
   SYSMAN> SET ENVIRONMENT/CLUSTER
   SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
   SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
   _SYSMAN> DISK$LOG2:[LOGFILES], DISK$LOG3:[LOGFILES]
   SYSMAN> EXIT
   ```

6. If you redefined SYS$JOURNAL in step 5, edit the SYS$STARTUP:SYLOGICALS.COM command procedure to update the SYS$JOURNAL definition.

   If you have created node-specific versions of SYLOGICALS.COM, edit all the versions.

7. Archive OLDNOD's transaction log.

8. Shut down OLDNOD, using the following command:

   ```
   $ @SYS$SYSTEM:SHUTDOWN.COM
   ```

   The shutdown procedure prompts you with a series of questions. Specify that an automatic reboot is not performed.

   For instructions on how to answer the questions, see Section 4.8.1.

9. Log in to another node.

10. Enable the following privileges: BYPASS, CMKRNL, NETMBX, OPER, SYSPRV, and VOLPRO.

11. Remove OLDNOD from the VMScluster, using the SYS$STARTUP:CLUSTER_CONFIG.COM command procedure, and reconfigure the VMScluster.

    For information on the CLUSTER_CONFIG.COM command procedure and on reconfiguring the VMScluster, see *VMScluster Systems for OpenVMS*.

Example

In this example, the VMScluster members and the locations of their transaction logs are as follows:

Node	Directory Containing Log
ORANGE	DISK$LOG1:[LOGFILES]
RED	DISK$LOG2:[LOGFILES]
WHITE	DISK$LOG3:[LOGFILES]

The following instructions show how to remove node ORANGE. Assume that you do not have the SETPRV privilege, and that none of the nodes have node-specific versions of the SYLOGICALS.COM command procedure.

First, log in to node ORANGE and enter the following command:

```
$ SET PROCESS/PRIVILEGES=(CMKRNL,EXQUOTA,LOG_IO,OPER,SYSNAM,SYSPRV,TMPMBX,WORLD)
```

Next, stop all software on node ORANGE that uses DECdtm services, then enter the following commands:

```
$ RUN SYS$SYSTEM:LMCP
LMCP> DUMP/ACTIVE SYSTEM$ORANGE:LM$JOURNAL

Dump of log file DISK$USER1:[LOGFILES]SYSTEM$ORANGE.LM$JOURNAL
    .
    .
    .
Total of 0 transactions active, 0 prepared and 0 committed.
LMCP> EXIT
```

The transaction log contains no active transactions, so it is safe to continue. Now enter the following commands:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEFINE/SYSTEM/EXECUTIVE SYS$JOURNAL -
_SYSMAN> DISK$LOG2:[LOGFILES], DISK$LOG3:[LOGFILES]
SYSMAN> EXIT
```

Next, edit the SYS$STARTUP:SYLOGICALS.COM command procedure to update the SYS$JOURNAL definition, then archive the transaction log DISK$USER1:[LOGFILES]SYSTEM$ORANGE.LM$JOURNAL.

Now enter the following command:

```
$ @SYS$SYSTEM:SHUTDOWN.COM
    .
    .
    .
Should an automatic system reboot be performed [NO]? NO
```

Finally, log in to either node RED or node WHITE and enter the following commands:

```
$ SET PROCESS/PRIVILEGES=(BYPASS,CMKRNL,NETMBX,OPER,SYSPRV,VOLPRO)
$ @SYS$STARTUP:CLUSTER_CONFIG.COM

       Cluster Configuration Procedure

            1. ADD a node to a cluster.
            2. REMOVE a node from the cluster.
            3. CHANGE a cluster member's characteristics.
            4. CREATE a duplicate system disk for BLUE.

    Enter choice [1]: 2
    .
    .
    .
 Updating network database...
 The configuration procedure has completed successfully.
```

23.10 Disabling DECdtm Services

DECdtm services start up automatically when you boot the system. The DECdtm services process, TP_SERVER, then checks for the existence of a transaction log on the system, and continues checking every 15 seconds until it finds one.

If you do not use, and do not plan to use, any software that uses DECdtm services, you can save memory and CPU time by disabling DECdtm services. This stops the creation of the TP_SERVER process when the system boots.

How to Perform This Task

To disable DECdtm services, edit the SYS$STARTUP:SYLOGICALS.COM command procedure to include the following:

```
$ !
$ DEFINE/SYSTEM/EXECUTIVE SYS$DECDTM_INHIBIT yes
$ !
```

If you have created node-specific versions of SYLOGICALS.COM, edit all the versions.

DECdtm services are disabled the next time you boot the system.

23.11 Enabling DECdtm Services

You need to enable DECdtm services only if you have previously disabled them and you now want to run software that uses them.

How to Perform This Task

1. Log in to any node in the VMScluster.
2. Enable the OPER privilege.
3. Deassign the logical name SYS$DECDTM_INHIBIT and start up DECdtm services using SYSMAN:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SET ENVIRONMENT/CLUSTER
SYSMAN> SET PROFILE/PRIVILEGES=SYSNAM
SYSMAN> DO DEASSIGN/SYSTEM/EXECUTIVE SYS$DECDTM_INHIBIT
SYSMAN> DO @SYS$STARTUP:DECDTM$STARTUP.COM
SYSMAN> EXIT
```

4. Edit the SYS$STARTUP:SYLOGICALS.COM command procedure to delete the SYS$DECDTM_INHIBIT definition. This ensures that DECdtm services start automatically when you boot the system.

24

Managing Special Processing Environments

The OpenVMS operating system supports the following special environments:

- Symmetric multiprocessing
- VAX Vector processing (VAX systems only, available on certain computers) ♦

This chapter describes how to set up and manage these special processing environments.

Information Provided in This Chapter

This chapter describes the following tasks:

Task	Section
Creating the multiprocessing environment	Section 24.2.1
Monitoring the multiprocessing environment	Section 24.2.2
†Loading vector processing support code	Section 24.4.1
†Configuring a vector processing system	Section 24.4.2
†Managing vector processes	Section 24.4.3
†Restricting access to the vector processor with ACLs	Section 24.4.4
†Obtaining information about a vector processing system	Section 24.4.5
†Loading the VAX Vector Instruction Emulation facility (VVIEF)	Section 24.4.6

†VAX specific

This chapter explains the following concepts:

Concept	Section
Symmetric multiprocessing	Section 24.1
Primary and secondary processors	Section 24.1.1
Available and active sets	Section 24.1.2
†Vector processing	Section 24.3
†VAX support for vector processing	Section 24.3.1
†The VAX Vector Instruction Emulation facility (VVIEF)	Section 24.3.2

†VAX specific

24.1 Understanding Multiprocessing

A multiprocessing system consists of two or more CPUs that address a common pool of memory and that are capable of executing instructions simultaneously.

The OpenVMS operating system supports a tightly coupled, symmetric multiprocessing (SMP) system. In a tightly coupled SMP system, all processors execute a single copy of the operating system and have equal access to all operating system code and system resources. As a result, in a tightly coupled SMP system, the scheduler can assign a job to any processor on the basis of which processor is free to execute the job, regardless of the requirements of the job or the system resources it must access. This capability is known as **dynamic load leveling**.

A multiprocessing system can function as an isolated entity, a node in a network, or a member of a VMScluster environment. Multiprocessing and uniprocessing systems run the same operating system, although multiprocessing can be enabled only on selected VAX and AXP processors. All processors in a multiprocessing environment must be at the same hardware and firmware level to guarantee that a given processor is capable of resuming the execution thread of a process that had been executing previously on another processor in the system.

24.1.1 Primary and Secondary Processors

In a multiprocessing system, one processor has the responsibility of starting other processors in the system. The **primary processor** is that processor in the system that is either logically or physically attached to the console device. As such, it is the processor that is the target of the console commands that boot the multiprocessing system. In this role, only the primary processor performs the initialization activities that define the operating system environment and prepare memory for the entire system. In addition, the primary processor serves as the system timekeeper, maintaining the system time and monitoring the timer queue for the expiration of its elements. In this sense, all processors in a multiprocessing system that do *not* have these responsibilities are known as **secondary processors**.

24.1.2 Available and Active Sets

An **available set** is made up of the processors that have passed the system's power-on hardware diagnostics and may or may not be actively involved in the system. Together, the primary and the secondary processors comprise the multiprocessing system's active set. The **active set** is the subset of the VAX or AXP system's processors that have passed its power-on diagnostics and are actively participating in system operations. The operating system identifies each processor in these sets by its **CPU ID**, a value prevalent in the syntax and displays of certain DCL and utility commands.

AXP

On AXP systems, CPU ID is also the value returned by interrogating the WHAM1 internal processor register. ♦

24.1.3 Processor Capabilities

The processors in a multiprocessing system offer certain capabilities to the processes executing in the system. The following capabilities are supported:

- Primary
- Quorum
- Run

- Vector

In addition, there are mechanisms to add and subtract other capabilities.

The Run capability affects CPU starting and stopping operations.

24.2 Managing Symmetric Multiprocessing (SMP) Environments

The following sections describe the tasks for managing multiprocessing systems.

24.2.1 Creating the Multiprocessing Environment

You can control the membership and character of a multiprocessing system at boot time by setting system parameters designed for these purposes. Among the system parameters that manage a multiprocessing system are the following:

Parameter	Function
MULTIPROCESSING	Determines which synchronization image is loaded into the operating system at boot time
SMP_CPUS	Determines which processors are brought into the multiprocessing environment from the available set at boot time

For more information about these and other system parameters, see the *OpenVMS System Management Utilities Reference Manual.*

You can add an available processor to the active set at boot time, or you can add it later using the DCL command START/CPU. The DCL command STOP/CPU removes a processor from the active set.

24.2.2 Monitoring the Multiprocessing Environment

Several operating system features provide special information about the character, capabilities, and status of a multiprocessor system. They include the DCL command SHOW CPU and the Monitor utility.

24.2.2.1 Obtaining Information About a Multiprocessor Configuration

The SHOW CPU command displays three levels of information describing the configuration and status of a multiprocessing system:

Level	Command Example	Display Contents
Summary	SHOW CPU	Indicates which processor is primary, which are configured, and which are active; displays the minimum revision levels for processors in the system and the setting of the MULTIPROCESSING system parameter; and indicates whether multiprocessing is enabled.
Brief	SHOW CPU/BRIEF	Produces information from the summary display; lists the current CPU state and the current process (if any) for each configured processor.
Full	SHOW CPU/FULL	Produces information from the summary display; lists the current CPU state, current process (if any), revision levels, and capabilities for each configured processor; indicates which processes can be executed only on certain processors.

For more information about the DCL commands relating to SMP, see the *OpenVMS DCL Dictionary*; for information about the Monitor utility, see the

MONITOR section in the *OpenVMS System Management Utilities Reference Manual*.

24.3 Understanding Vector Processing (VAX Only)

VAX

A single data item, having one value, is known as a **scalar**. A group of related scalar values, or elements, all of the same data type, is known as a **vector**.

Traditional (scalar) computers operate only on scalar values, and must process vector elements sequentially. Vector computers, on the other hand, recognize vectors as native data structures and can operate on an entire vector with a single vector instruction. Because this type of processing involves the concurrent execution of multiple arithmetic or logical operations, a vector computer can routinely process a vector four to five times faster than a traditional computer can using only scalar instructions.

Vector processors gain a further speed advantage over scalar processors by their use of special hardware techniques designed for the fast processing of streams of data. These techniques include data pipelining, chaining, and other forms of hardware parallelism in memory and in arithmetic and logical functional units. Pipelined functional units allow the vector processor to overlap the execution of successive computations with previous computations.

24.3.1 VAX Support for Vector Processing (VAX Only)

The VAX vector architecture includes sixteen 64-bit vector registers (V0 through V15), each containing 64 elements; vector control registers, including the vector count register (VCR), vector length register (VLR), and vector mask register (VMR); vector functional units; and a set of vector instructions. VAX vector instructions transfer data between the vector registers and memory, perform integer and floating-point arithmetic, and execute processor control functions. A more detailed description of the VAX vector architecture, vector registers, and vector instructions appears in the *VAX MACRO and Instruction Set Reference Manual*.

Those VAX systems that comply with the VAX vector architecture are known as **vector-capable** systems.

A VAX vector processing system configuration includes one or more integrated scalar-vector processor pairs, or **vector-present processors**. Such a configuration can be symmetric, including a vector coprocessor for each scalar, or asymmetric, incorporating additional scalar-only processors. Depending upon the model of the VAX vector processing system, the scalar and vector CPUs of vector-present processors can be either a single, integral physical module or separate, physically independent modules. In either case the scalar and vector CPUs are logically integrated, sharing the same memory and transferring data over a dedicated, high-speed internal path. Because the CPUs are thus tightly coupled, use of the vector CPU eliminates the expense of I/O operations.

Like VAX scalar processing systems, a VAX vector processing system can participate as a member of a VAXcluster or a node in a network, or be run as a standalone system.

24.3.2 The VAX Vector Instruction Emulation Facility (VVIEF) (VAX Only)

The VAX Vector Instruction Emulation facility (VVIEF) is a standard feature of the OpenVMS operating system that allows vectorized applications to be written and debugged in a VAX system in which vector processors are not available. VVIEF emulates the VAX vector processing environment, including the nonprivileged VAX vector instructions and the vector system services. Use of VVIEF is restricted to user mode code.

VVIEF is strictly a program development tool, and *not* a run-time replacement for vector hardware. There is no performance benefit from vectorizing applications to run under VVIEF; vectorized applications running under VVIEF will execute more slowly than their scalar counterparts.

the operating system supplies the VVIEF bootstrap code as an executive loadable image. You invoke the command procedure SYS$UPDATE:VVIEF$INSTAL.COM to cause the system to load VVIEF at the *next* system boot and each successive system boot. Note that, in the presence of OpenVMS vector support code, VVIEF remains inactive. Although it is possible to prevent the loading of vector support code in a vector-present system (see Section 24.4.1) and activate VVIEF, there are few benefits. Should the only scalar-vector processor pair in the system fail, the execution of preempted vectorized applications will not be resumed under the emulator.

See Section 24.4.6 for additional information on loading and unloading VVIEF. ♦

24.4 Managing the Vector Processing Environment (VAX Only)

VAX

The following sections describe these tasks for managing a vector processing system.

24.4.1 Loading the Vector Processing Support Code (VAX Only)

By default, in a VAX vector processing system, the system automatically loads the vector processing support code at boot time. You can override the default behavior by setting the static system parameter VECTOR_PROC as described in Table 24–1.

Table 24–1 Settings of VECTOR_PROC System Parameter

Value	Result
0	Do not load the vector processing support code, regardless of the system configuration.
1	Load the vector processing support code if there is at least one vector-present processor. This is the default value.
2	Load the vector processing support code if the system is vector-capable. This setting is most useful for a system in which processors have separate power supplies. With this setting, you can reconfigure a vector processor into the system without rebooting the operating system.

24.4.2 Configuring a Vector Processing System (VAX Only)

You can add a vector-present processor to or remove it from a multiprocessing configuration at boot time by using the system parameter SMP_CPUS, or at runtime by using the DCL commands START/CPU and STOP/CPU. Note that the operating system treats the scalar and vector CPU components of a vector-present processor as a single processor, starting them and stopping them together.

At boot time, the setting of the system parameter SMP_CPUS identifies which secondary processors in a multiprocessing system are to be configured, including those processors that are vector present. (the operating system always configures the primary processor.) The default value of –1 boots all available processors, scalar and vector-present alike, into the configuration. (See the *OpenVMS System Management Utilities Reference Manual* for additional information on this parameter.) Note that, prior to starting a vector-present processor, you should ensure that the vector processing support code (see Section 24.4.1) is loaded at boot time. Otherwise, processes will be able to use only the scalar CPU component of the vector-present processor.

To bring secondary processors into a running multiprocessing system, you use the DCL command START/CPU. To remove secondary processors from the system, use the STOP/CPU commands. Again, you must ensure that the vector processing support code has been loaded at boot time for the vector CPU component of vector-present processors started in this way to be used.

Note, however, that a STOP/CPU command fails and generates a message if it would result in the removal of a vector-present processor that is the sole provider of the vector capability for currently active vector consumers. In extreme cases, such as the removal of a processor for repair, you can override this behavior by issuing the command STOP/CPU/OVERRIDE. This command stops the processor, despite stranding processes.

When a STOP/CPU/OVERRIDE command is issued for a vector-present processor, or when a vector-present processor fails, the operating system puts all stranded vector consumers into a CPU-capability-wait (RSN$_CPUCAP) state until a vector-present processor is returned to the configuration. To any other process that subsequently issue a vector instruction (including a marginal vector consumer), the system returns a “requested CPU not active” message (CPUNOTACT).

See the *OpenVMS DCL Dictionary* for additional information on the START/CPU and STOP/CPU commands.

24.4.3 Managing Vector Processes (VAX Only)

The operating system scheduling algorithms automatically distribute vector and scalar processing resources among vector consumers, marginal vector consumers, and scalar consumers. However, VAX vector processing configurations vary in two important ways:

- The amount of vector processing activity the configuration must accommodate
- The number of vector-present processors that are available in the configuration to service vector processing needs

In a configuration in which there are more vector consumers in a system than there are scalar-vector processor pairs to service them, vector consumers share vector-present processors according to process priority. At a given priority, the system schedules vector consumers on a vector-present processor in a round-robin fashion. Each time the system must schedule a new vector consumer on a vector-present processor, it must save the vector context of the current vector consumer in memory and restore the vector context of the new vector consumer from memory. When such “slow” vector context switches occur too frequently, a significant portion of the processing time is spent on vector context switches relative to actual computation.

Systems that have heavy vector processing needs should be adequately configured to accommodate those needs. There are, however, some mechanisms by which you can tune the performance of an existing configuration.

24.4.3.1 Adjusting System Resources and Process Quotas (VAX Only)

Systems in which several vector consumers are active simultaneously may experience increased paging activity as processes share the available memory. To reduce process paging, you may need to use the Authorize utility to adjust the working set limits and quotas of the processes running vectorized applications. (See the AUTHORIZE section of the *OpenVMS System Management Utilities Reference Manual* for additional information.) An increase of the process maximum working set size (system parameter WSMAX) may also be necessary. Additionally, a vectorized application may use the Lock Pages in Working Set system service (SYS$LKWSET) to enhance its own performance.

The system allots to each vector consumer 8 KB of system nonpaged dynamic memory in which the operating system stores vector context information. Depending upon how many vector consumers may be active in the system simultaneously, you may need to adjust the system parameter NPAGEDYN. The DCL command SHOW MEMORY/POOL/FULL displays the current size of nonpaged pool in bytes.

To obtain optimal performance of a VAX vector processing system, you should take some care in setting up generic batch queues that avoid saturating the system's vector resources. If a queue contains more active vectorized batch jobs than there are vector-present processors in the system, a significant portion of the processing time will be spent on vector context switches.

The recommended means for dispatching vectorized batch jobs to a VAX vector processing system is to set up a separate queue (for instance, VECTOR_BATCH) with a job limit equal to the number of vector-present processors in the system. When submitting vectorized batch jobs, users should be encouraged to submit them to this generic vector-processing batch queue.

24.4.3.2 Distributing Scalar and Vector Resources Among Processes (VAX Only)

As a vector consumer, a process must be scheduled only on a vector-present processor. If the image the process is executing issues only scalar instructions for a period of time, and it must share the scalar-vector processor pair with other vector consumers, its inability to run on an available scalar processor could hamper its performance and the overall performance of the system.

By default, the operating system assumes that if a vector consumer has not issued a vector instruction for a certain period of time, it is unlikely that it will issue a vector instruction in the near future. The system relinquishes this process's need for the vector capability, classifying it as a marginal vector consumer.

In an asymmetric vector-processing configuration, detection of marginal vector consumers achieves the following desirable effects:

- Because a marginal vector consumer is eligible to run on a larger set of processors, its response time will improve.
- The scheduling of marginal vector consumers on scalar processors reduces the contention for vector-present processors.
- Because vector consumers issuing vector instructions are more likely to be scheduled on vector-present processors, the vector CPU is more efficiently used.

Use the VECTOR_MARGIN system parameter to establish the interval of time at which the system checks the status of all vector consumers. The VECTOR_MARGIN parameter accepts an integer value between 1 and $FFFFFFFF_{16}$. This value represents a number of consecutive process quanta (as determined by the system parameter QUANTUM). If the process has not issued any vector instructions in the specified number of quanta, the system declares it a marginal vector consumer.

The default value of the VECTOR_MARGIN parameter is 200_{10}.

24.4.4 Restricting Access to the Vector Processor by Using ACLs (VAX Only)

A vector **capability** is a software abstract by which the operating system makes the services of the vector processor available to users. You can restrict the use of the vector processor to users holding a particular identifier by associating an access control list (ACL) with the vector capability object.

For example, a university might limit use of the vector processor to faculty and students in an image processing course, or a service bureau might charge users for access to the vector capability, time spent on the vector processor, or both.

Use the DCL command SET ACL in the following format to establish access control entries (ACEs) on a vector capability:

SET ACL/OBJECT=CAPABILITY VECTOR/ACL[=(ace[,...])])

Note that you must be in the SYSTEM user category (as described in the *OpenVMS User's Manual*) to set an ACL on the vector capability.

The following DCL command displays the ACL on the vector capability:

```
$ SHOW ACL/OBJECT=CAPABILITY VECTOR
```

Note that the ACL is on the vector capability, not on the use of any or all vector-present processors in the system. The operating system will still schedule processes without permission to use the vector capability on a vector-present processor. However, these processors will be able to use only the scalar CPU component of the processor, and cannot execute vector instructions. Likewise, because the ACL is on the vector capability and not on a vector-present processor, you cannot establish an ACL to force long-running jobs to a specific processor.

For additional information on the SET ACL and SHOW ACL commands, see the *OpenVMS DCL Dictionary*.

24.4.5 Obtaining Information About a Vector Processing System (VAX Only)

You can obtain information about the status of the vector processing system and the use of the system by individual processes through various means, including:

- The DCL lexical functions F$GETJPI and F$GETSYI
- The DCL command SHOW CPU
- The DCL commands SHOW PROCESS and LOGOUT/FULL
- The Accounting utility
- The Monitor utility

24.4.5.1 DCL Lexical Functions F$GETJPI and F$GETSYI (VAX Only)

The DCL lexical function F$GETJPI accepts the following items and returns the corresponding information regarding the vector status of a specified process:

Item	Return Type	Information Returned
FAST_VP_SWITCH	Integer	Number of times this process has issued a vector instruction that resulted in an inactive vector processor being enabled without the expense of a vector context switch
SLOW_VP_SWITCH	Integer	Number of times this process has issued a vector instruction that resulted in an inactive vector processor being enabled with a full vector context switch
VP_CONSUMER	Boolean	Flag indicating whether the process is a vector consumer
VP_CPUTIM	Integer	Total amount of time the process has accumulated as a vector consumer

The DCL lexical function F$GETSYI accepts the following items and returns the corresponding information regarding the status of the vector processing system:

Item	Return Type	Information Returned
VECTOR_EMULATOR	Integer	Flag indicating the presence of the VAX Vector Instruction Emulation facility (VVIEF) in the system
VP_MASK	Integer	Mask indicating which processors in the system have vector coprocessor
VP_NUMBER	Integer	Number of vector processors in the system

See the *OpenVMS DCL Dictionary* for additional information about the DCL lexicals F$GETJPI and F$GETSYI.

24.4.5.2 SHOW CPU/FULL Command (VAX Only)

The SHOW CPU/FULL command lists the capabilities of the specified CPU. Issue this command to determine the presence of the vector capability in the system prior to executing a STOP/CPU command.

See the *OpenVMS DCL Dictionary* for additional information about the SHOW CPU command.

24.4.5.3 SHOW PROCESS and LOGOUT/FULL Commands (VAX Only)

If the target process has accrued any time as a vector consumer scheduled on a vector-present processor, the DCL commands SHOW PROCESS and LOGOUT /FULL display the elapsed vector CPU time and the charged vector CPU time, respectively.

To accumulate vector CPU time, a process must be a vector consumer (that is, require the system vector capability) and be scheduled on a vector-present processor. The operating system still charges the vector consumer vector CPU time, even if, when scheduled on the vector-present processor, it does not actually use the vector CPU. Note that, because scalar consumers and marginal vector consumers do not use the vector CPU, they do not accrue vector CPU time, even when scheduled on a vector-present processor.

See the *OpenVMS DCL Dictionary* for additional information about the SHOW PROCESS and LOGOUT commands.

24.4.6 Loading the VAX Vector Instruction Emulation Facility (VVIEF) (VAX Only)

The VAX Vector Instruction Emulation facility (VVIEF) is a standard operating system feature that allows vectorized applications to be written and debugged in a VAX system in which vector processors are not available. VVIEF is intended strictly as a program development tool, and *not* as a run-time replacement for vector hardware. There is no performance benefit from vectorizing applications to run under VVIEF; vectorized applications running under VVIEF will execute more slowly than their scalar counterparts.

The operating system supplies the VVIEF bootstrap code as an executive loadable image. To cause the system to load VVIEF at the *next* system boot and at each subsequent system boot, invoke the command procedure SYS$UPDATE:VVIEF$INSTAL.COM. To unload VVIEF, invoke the command procedure SYS$UPDATE:VVIEF$DEINSTAL.COM and reboot the system. You can determine the presence or absence of VVIEF in a system by issuing the following DCL commands:

```
$ X = F$GETSYI("VECTOR_EMULATOR")
$ SHOW SYMBOL X
X = 1   Hex = 00000001  Octal = 0000000001
```

A return value of 1 indicates the presence of VVIEF; a value of 0 indicates its absence.

Note that, although VVIEF may be loaded into the system, in the presence of vector support code, it remains inactive. Although it is possible to prevent the loading of vector processing support code in a vector-present system (see Section 24.4.1) and activate VVIEF, there are few benefits. Should the only vector-present processor in the system fail, the execution of preempted vectorized applications will not resume under the emulator. ♦

A
Files–11 Disk Structure

This appendix explains disk terminology and disk concepts. It also describes reserved files, points out those files used by the Analyze/Disk_Structure utility (ANALYZE/DISK_STRUCTURE), and compares Files–11 On-Disk Structure (ODS) Level 1 and Files–11 ODS Level 2.

A.1 Disk Concepts

This section defines terms related to both the physical and the logical organization of disks.

A.1.1 Logical Organization of a Disk

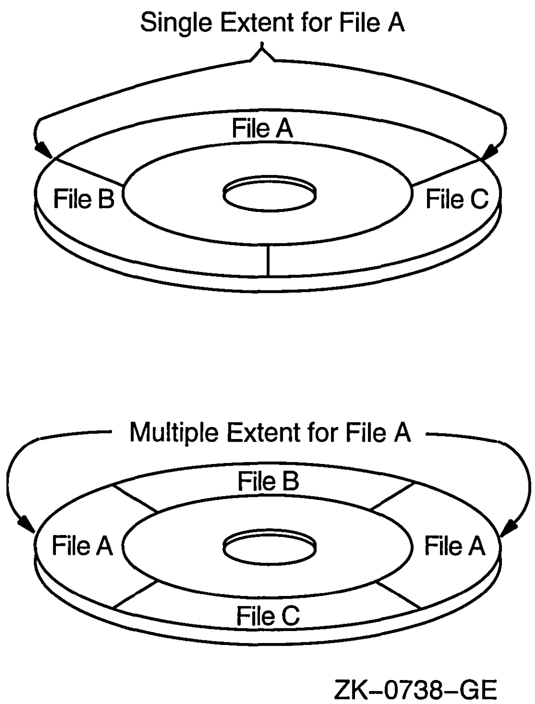
The smallest addressable unit of information on a disk is a **block**. Files–11 On-Disk Structures define a block to consist of 512 8-bit bytes. Blocks can be treated as units for transfer between a Files–11 disk volume and memory. Files–11 ODS, however, views a disk as an array of blocks, and is generally not concerned with individual blocks.

Blocks are logically grouped into **clusters**, which are the basic units by which disk space is allocated. You determine the number of blocks in a cluster when a given disk, known as a **volume**, is first prepared for use (initialized). Cluster sizes vary for different media types. The smaller cluster sizes in the range are usually more practical. In general, a disk with a relatively small number of blocks is given a smaller cluster size, while larger disks are given larger cluster sizes to minimize the overhead for disk space allocation.

Contiguous clusters allocated to a particular file are called **extents**. An extent can contain all or part of a file. If enough contiguous area is available on the disk, the entire file is allocated as a single extent. Sometimes, however, not enough contiguous area is available to hold the entire file, or, when you create a file initially, you might not want to reserve the entire required amount of space. When the file is eventually extended, it is unlikely that the adjacent clusters will still be unallocated. If the adjacent clusters are already allocated to another file, the extension does not occur contiguously.

If a file is divided into two or more parts, each part is an extent. Thus, a file can consist of multiple extents located in separate areas on the disk, as shown in Figure A–1. Note that the file extensions are done automatically.

Figure A–1 File Extents



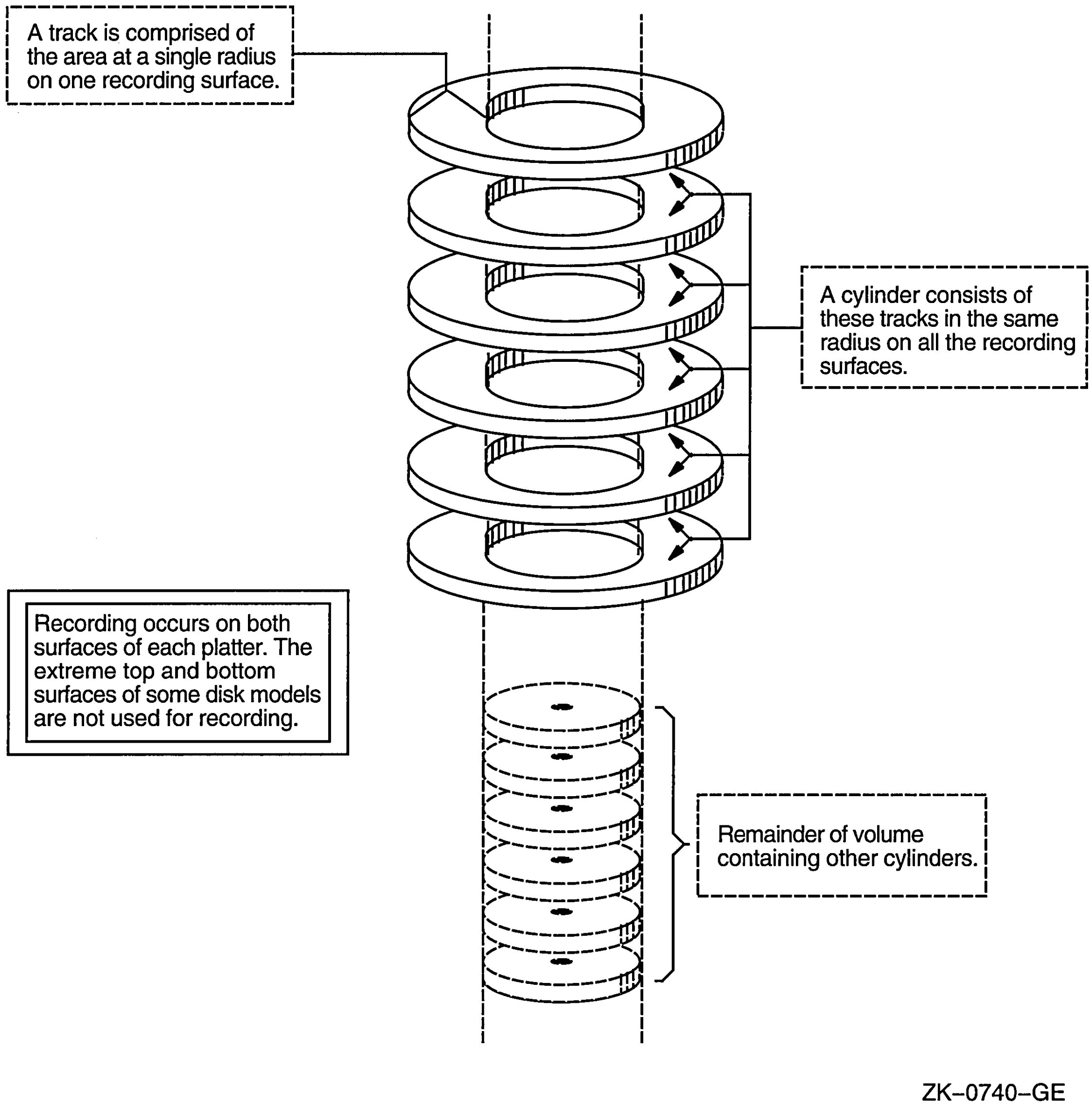
A.1.2 Physical Organization of a Disk

The smallest unit discernible to the Files–11 structure is the **sector**; for most Files–11 disks, a sector is equivalent to a block, which is 512 bytes. Other basic terms related to disks are **track** and **cylinder**. A track is the collection of sectors (or blocks, on Files–11 structures) at a single radius on one recording surface of a disk. It is accessible to a given read/write head position on the disk device. A cylinder consists of all tracks at the same radius on all recording surfaces of a disk.

Because access to any of the blocks in a given cylinder does not require any movement of the disk's read/write heads, it is generally advantageous to keep related data blocks in the same cylinder. For this reason, when choosing a cluster size for a large-capacity disk, you should usually select a cluster size that divides evenly into the cylinder size.

Figure A–2 is a graphic representation of disk tracks and cylinders.

Figure A–2 Tracks and Cylinders



A.2 Files–11 Structure

The Files–11 structure creates a set of nondeletable reserved files when a volume or volume set is initialized. These files control the organization of a Files–11 disk. A Files–11 structure resides on a volume, which is a physical medium such as a disk pack. A Files–11 volume is an ordered set of 512-byte blocks. The blocks are numbered consecutively from 0 to *n–1*; the value of *n–1* is the size of the disk in blocks.

A.2.1 File Identification (FID)

Each file on a Files–11 disk is identified by a unique, system-assigned file identification (FID) and can have a user-assigned alphanumeric name. The primary function of a Files–11 directory is to associate the user-assigned alphanumeric name of each file with the unique FID of the file. This association ensures that files present on a volume are retrievable by name.

The FID of a file consists of a set of three numbers. The first is the **file number** (NUM). The file system uses this number as an offset into the index file (reserved file INDEXF.SYS), which stores information for all files on a volume.

The second part of the FID is the **file sequence number** (SEQ), which represents the number of times a particular file number has been used. File numbers are allocated and deallocated as files are created and deleted. As a result, the file number alone cannot uniquely identify the file. By incrementing the sequence number each time a file number is used, the file system ensures that each file has a unique identification in INDEXF.SYS.

The third number in the FID is the **relative volume number** (RVN). This number indicates the volume (of a volume set) on which the file resides (ODS–2 only). If the volume set consists of a single volume, the RVN of all files on that volume is 1.

A.2.2 ODS Directory Hierarchies

The Files–11 ODS–2 structure is a multilevel directory hierarchy. The top level of the directory structure is the master file directory (MFD). The MFD of a volume is always named [000000]. The MFD contains all top-level directories, including itself, and reserved files.

A directory is a file that contains other files. A file contained in a directory can also be a directory and contain other files. By nesting directories, users can construct directory hierarchies up to eight levels deep (including the master file directory).

In a volume set, the MFD for all of the user directories on the volume set is located on relative volume 1. The entries of this MFD point to directories located on any volume in the set. These directories in turn point to files and subdirectories on any volume in the set. The MFD of any remaining volume in the set includes only the names of the reserved files for that volume.

VAX

On VAX systems, the Files–11 ODS–1 structure supports a two-level directory hierarchy. Each user identification code (UIC) is associated with a user file directory (UFD). Each UFD is included in the master file directory (MFD) of the volume. ♦

A.3 Reserved Files

This section describes the reserved files that Files–11 uses. Note that all reserved files have constant FIDs.

This section also points out the files ANALYZE/DISK_STRUCTURE uses. ANALYZE/DISK_STRUCTURE makes an in-memory copy of what these files should look like and compares it with the current version. The utility reports and repairs (if you specify the /REPAIR qualifier) any discrepancies found during these comparisons.

Table A–1 shows the reserved files used by Files–11 Level 1 and Level 2, and files used by ANALYZE/DISK_STRUCTURE.

Table A–1 Reserved Files

Reserved File	File Name	†Structure Level 1	Structure Level 2	ANALYZE/ DISK_ STRUCTURE
Index file	INDEXF.SYS;1	X	X	X
Storage bit map file	BITMAP.SYS;1	X	X	X
Bad block file	BADBLK.SYS;1	X	X	
Master file directory	000000.DIR;1	X	X	X
Core image file	CORIMG.SYS;1	X	X	
Volume set list file	VOLSET.SYS;1		X	X
Continuation file	CONTIN.SYS;1		X	
Backup log file	BACKUP.SYS;1		X	
Pending bad block	BADLOG.SYS;1		X	
Quota file	QUOTA.SYS			X
†Volume security profile	SECURITY.SYS		X	

†VAX specific

A.3.1 Index File, INDEXF.SYS

Every Files–11 volume has an index file, which is created when the volume is initialized. (You cannot use a disk as a Files–11 disk until it has been initialized with the INITIALIZE command.)

INDEXF.SYS is a large, extendable file made up of several sections. These sections provide the operating system with the information necessary to identify a Files–11 volume, initially access that volume, and locate all the files on that volume (including INDEXF.SYS itself).

Table A–2 shows the information that is in INDEXF.SYS. Following the table are additional explanations of boot block, home block, and file headers.

Table A–2 Contents of Files–11 Index File

Term	Definition
Boot block	Virtual block number 1 of the index file. If the volume is a system volume, the boot block contains a boot program that loads the operating system into memory. If the volume is not a system volume, the boot block contains a program that displays the message that the volume is not the system device but a device that contains users' files only.
Home block	Establishes the specific identity of the volume, providing such information as the volume name and protection, the maximum number of files allowed on the volume, and the volume ownership information. The home block is virtual block number 2 of the index file.
Backup home block	A copy of the home block; permits the volume to be used even if the primary home block is destroyed.

(continued on next page)

Table A–2 (Cont.) Contents of Files–11 Index File

Term	Definition
Backup index file header	Permits data on the volume to be recovered if the index file header is corrupted; occupies virtual blocks *v* * 3 + 1 through *v* * 4, where *v* is the volume cluster factor.
Index file bitmap	Controls the allocation of file headers and thus the number of files on the volume; contains a bit for each file header allowed on the volume. If the value of a bit for a given file header is 0, a file can be created with this file header. If the value is 1, the file header is already in use.
File headers	Makes up the bulk of the index file; contain all the information needed for gaining access to the file. Each file header describes one file on the volume. A file header contains information such as the owner UIC, protection code, creation date and time, and access control lists (ACLs); it also contains a list of extents that make up the file, describing where the file is logically located on the volume. Note that a file header can also be an extension header.
Alternate index file header	Permits recovery of data on the volume if the primary index file header becomes damaged.

A.3.1.1 Boot Block

Block 0 on a system disk is the **boot block**. It contains the location and size of the **bootstrap image**, which is used to boot the system. Certain processors, in order to boot, must read this boot block to obtain the location of the bootstrap image. For more details, see Section 4.6.

A.3.1.2 Home Block

The **home block** is normally the next block after the boot block; it identifies the disk as a Files–11 volume. If for some reason the home block cannot be read (physically unusable), an alternative block will be selected for use as the home block. This block provides specific information about the volume and default values for files on the volume. Among the items in the home block are the following:

- The volume name
- Information to locate the remainder of the index file
- The maximum number of files that can be present on the volume at any one time
- The user identification code (UIC) of the owner of the volume
- Volume protection information (specifies which users can read or write the entire volume)

Files–11 volumes contain several copies of the home block to ensure against accidental destruction of this information and the consequent loss of access to files on the volume.

A.3.1.3 File Headers

Most of the index file consists of **file headers**; each file header describes a portion of a file on the volume. File headers contain information such as the owner UIC, protection code, creation date and time, and access control lists (ACLs). Most importantly, the file header contains a list of extents that make up the file, describing where the file is logically located on the volume. If a file has a large number of extents, multiple file headers may be used to describe them. A file identifier number is associated with each file header.

When you create a file, you normally specify a file name to OpenVMS RMS, which assigns this name to the file on a Files–11 volume. OpenVMS RMS places the file name and file identifier associated with the newly created file into a directory, which contains an entry defining the location for each file. When you access the file, you supply the file name, which supplies a path to the file identifier through the directory entry. The file identifier, in turn, points to the location of the file header, which contains a listing of the extent or extents that locate the actual data.

Because they represent the current state of file storage on a volume, file headers are of particular interest to ANALYZE/DISK_STRUCTURE. Each file on a Files–11 disk (INDEXF.SYS included) is identified and located by a primary header (and extension headers, if required) in INDEXF.SYS.

Each fixed-length header contains both constant and variable-length data. This data is stored in one of the six areas shown in Table A–3.

Table A–3 Areas of Data in File Headers

Area of Data	Description
Header	This area contains the header identification, the file number and its sequence number, the protection code for the file, and offsets to the other file header areas.
Ident	This area contains the identification and accounting data for the file (for example, the name of the file, its creation date and time, and backup date and time).
Map	This area contains a list of retrieval pointers that map the virtual blocks of the file to the logical blocks of the volume. Each pointer describes one group of consecutively numbered logical blocks that is allocated to the file. Retrieval pointers are arranged in the order of the virtual blocks they represent.
Access control list	An optional area that contains ACL-related information.
Reserved	This area is reserved for use by special applications.
End checksum	The last two bytes of the file header contain a 16-bit additive checksum of the preceding 255 words of the file header. The checksum helps verify that the block is a valid file header.

A set of contiguous clusters is known as an **extent**. The size of an extent varies according to the number of contiguous clusters. For example, assume a file requires 1000 blocks of storage, and the file system finds a set of 800 contiguous blocks and a set of 200 contiguous blocks. The file would then be stored in two extents: one consisting of 800 blocks, the other of 200.

The **primary header** of a file points to the first extent of that file and to as many extents as can be stored in the map area of the primary header. When the number of extents required to contain a file exceeds the map area available in the primary header, or the ACL is too large to fit in the primary header, the file is allocated an **extension header**. Extension headers contain all the constant data of the primary header, as well as the variable data (in the header map area and access control list) that specifies the locations of the extents to which the extension header points.

ANALYZE/DISK_STRUCTURE confirms the validity of a file by working its way down the list of primary and extension headers of the file. During this process, ANALYZE/DISK_STRUCTURE checks the validity of the file header, the chain of pointers to all extension headers, the retrieval pointers in all headers, and the attributes of the file.

A.3.2 Storage Bit Map File, BITMAP.SYS

The storage bit map file is a contiguous file that the file system uses to keep track of the available space on a volume. This file contains a storage control block (SCB), which consists of summary information intended to optimize the Files–11 space allocation, and the bit map itself, which lists the availability of individual blocks.

The SCB contains summary information about the volume (cluster factor, volume size, blocking factor, and so forth). Each bit in the bitmap represents an allocatable cluster on the volume. If a bit is set, the corresponding cluster is available for use. If a bit is clear, the cluster is not available.

During normal operation, the operating system moves portions of the bitmap in and out of cache memory. The state of each bit in memory is altered as clusters are allocated and deallocated. BITMAP.SYS is updated when the portion of the bitmap in cache is swapped back to disk. Since there is always a portion of the bitmap in cache, BITMAP.SYS never reflects the current state of allocated clusters on a disk (unless the disk is dismounted or write-locked).

One of the functions of ANALYZE/DISK_STRUCTURE is to build a current version of BITMAP.SYS from data extracted from INDEXF.SYS, so that BITMAP.SYS accurately reflects the status of free clusters on the disk.

A.3.3 Bad Block File, BADBLK.SYS

The bad block file contains all the bad blocks on the volume. The system detects bad disk blocks dynamically and prevents their reuse once the files to which they are allocated have been deleted.

A.3.4 Master File Directory

The MFD is listed in the master file directory as 000000.DIR;1. The MFD, which is the root of the volume's directory structure, lists the reserved files that control the volume structure and may list both users' files and users' file directories.

Usually, however, the MFD is used to list the reserved files and users' file directories; users seldom enter files into the MFD, even on private volumes. In fact, on a private volume, it is most convenient for users to create a directory that has the same name as their default directory on a system disk. For an explanation of users' file directories and file specifications, see the *OpenVMS User's Manual.*

When the Backup utility (BACKUP) creates sequential disk save sets, it stores the save-set file in the MFD.

ANALYZE/DISK_STRUCTURE verifies all files contained in the directory structure by making comparisons to INDEXF.SYS. Any file found in INDEXF.SYS that is not traceable through the directory structure is "lost." ANALYZE/DISK_STRUCTURE places lost files in the top-level directory SYSLOST.DIR if you specified /REPAIR in the command.

A.3.5 Core Image File, CORIMG.SYS

The core image file is not used by the operating system.

A.3.6 Volume Set List File, VOLSET.SYS

The volume set list file is used only on relative volume 1 of a volume set. The file contains a list of the labels of all the volumes in the set and the name of the volume set.

ANALYZE/DISK_STRUCTURE uses VOLSET.SYS to locate each volume in the set and confirm the attributes of each volume. Since all volume set information is stored in VOLSET.SYS on relative volume 1, ANALYZE/DISK_STRUCTURE ignores VOLSET.SYS on all other volumes.

A.3.7 Continuation File, CONTIN.SYS

The continuation file is used as the extension file identifier when a file crosses from one volume to another volume of a loosely coupled volume set. This file is used for all but the first volume of a sequential disk save set.

A.3.8 Backup Log File, BACKUP.SYS

The backup log file is reserved for future use.

A.3.9 Pending Bad Block Log File, BADLOG.SYS

The pending bad block log file contains a list of suspected bad blocks on the volume that are not listed in the bad block file.

A.3.10 Quota File, QUOTA.SYS

The quota file is a reserved file that is used by the file system to keep track of the disk usage of each UIC on a volume. If you enable disk quota checking for a volume, the records of the file QUOTA.SYS contain all the UICs on the volume. The system constantly updates QUOTA.SYS to reflect the current disk usage, the maximum allowed disk usage, and the permitted overdraft for each UIC.

During the course of its operations, ANALYZE/DISK_STRUCTURE creates a version of QUOTA.SYS in memory that reflects the actual disk usage for each UIC. This version is eventually compared to the disk version of QUOTA.SYS. If ANALYZE/DISK_STRUCTURE detects any disparities in disk usage, ANALYZE/DISK_STRUCTURE notifies you. If you invoked ANALYZE/DISK_STRUCTURE with the /REPAIR qualifier, the disk version of QUOTA.SYS is updated.

A.3.11 Volume Security Profile, SECURITY.SYS (VAX Only)

VAX

The volume security profile includes the volume owner UIC, the volume system-owner-group-world (SOGW) protection mask, and the volume access control list (ACL). ♦

A.4 Files–11 ODS Level 1 Versus Level 2 (VAX Only)

VAX

On VAX systems, for reasons of performance, reliability, and security, Files–11 ODS Level 2, a compatible superset of ODS Level 1, is the preferred disk structure on the system. At volume initialization time, Structure Level 2 is the default. (See the INITIALIZE command in the *OpenVMS DCL Dictionary*.)

On VAX systems, specify ODS Level 1 only for volumes that must be transportable to RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems, as these systems support only that structure level. Additionally, you might be required to handle Structure Level 1 volumes transported to OpenVMS from one of these systems.

Where Structure Level 1 volumes are in use on the system, bear in mind the limitations on them that are shown in Table A–4.

Table A–4 Limitations on Files–11 Structure Level 1 Volumes

Disk	Only Files–11 ODS–2 disks are protected objects.
Directories	No hierarchies of directories and subdirectories, and no ordering of directory entries (that is, the file names) in any way. RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems do not support subdirectories and alphabetical directory entries.
Disk quotas	Not supported.
Multivolume files and volume sets	Not supported.
Placement control	Not supported.
Caches	No caching of file header blocks, file identification slots, or extent entries.
System disk	Cannot be a Structure Level 1 volume.
VMScluster access	Local access only; cannot be shared across a cluster.
Clustered allocation	Not supported.
Backup home block	Not supported.
Protection code E	E means “extend” for the RSX–11M operating system but is ignored by OpenVMS.
File versions	Limited to 32,767; version limits are not supported.
Enhanced protection features (for example, access control lists)	Not supported.
Long file names	Not supported.
RMS journaling	Not supported.
RMS execution statistics monitoring	Not supported.

Future enhancements to OpenVMS software will be based primarily on Structure Level 2; therefore, Structure Level 1 volumes might be further restricted in the future. ♦

Glossary

access control list (ACL)

A protection mechanism using a more refined level of protection than that available with **UIC-based protection**. ACLs can be used to grant or deny access to individual users or groups of users.

access mode

Any of the four processor access modes in which software executes. Processor access modes prevent system software from inadvertently performing operations that might damage the system. Processor access modes are, in order from most to least privileged and protected: kernel, executive, supervisor, and user. When the processor is in any mode other than kernel mode, the processor is inhibited from executing privileged instructions.

account

Each system user has an account. When you log in, you log in under a particular account name and number. This number informs the system where your files are and what kind of access to other files and system facilities you should be given.

accounting files

Files where the system stores information on resource use. Compare with **current accounting file**.

active set

In a multiprocessing system, the subset of processors that have successfully run power-on diagnostics and are actively participating in system operations. Compare with **available set**.

active values

With system parameters, the set of values that is stored in memory and is used by the active system. When the system boots, it reads into memory the **current values** stored in a parameter file on disk.

adjacent node

In a network, a node that is connected to your node by a single physical line.

allocation class

In a VMScluster environment, for devices that are dual-ported between two computers, a numeric value used to create a unique, path-independent device name.

answer file

A file in the form SYS$UPDATE:*product*.ANS. The file is created when you install a product initially, and you specify the Auto-Answer option. The file contains a record of the answers you entered when you ran VMSINSTAL.COM to install that product initially.

application service

A **LAT service** in which LAN users can access only a specific program. Contrast with **general timesharing service**.

area router

In a network, a node that performs routing operations between areas and within its own area. Also called a **level 2 router**. Compare with **level 1 router**.

autostart feature

A feature that simplifies startup and ensures high availability of execution queues in a VMScluster environment. It lets you do the following:

- Start all **autostart queues** on a node with a single command
- Specify a list of nodes (within a VMScluster environment) to which a queue can automatically fail over if necessary.

autostart queue

An execution queue that takes advantage of the **autostart feature**. When you create a queue, you can designate it as an autostart queue.

available set

In a multiprocessing system, those processors that have successfully completed the system's power-on hardware diagnostics and may or may not be actively involved in the system. Compare with **active set**.

backlink

In Files–11 disk structure, a pointer to the directory in which a file resides.

banner page

A specially formatted page that prints at the beginning and end of print jobs and files within print jobs. These pages are helpful in identifying and separating output jobs, and the files within those jobs, when they are printed.

base process priority

A base priority value that the system uses to schedule a process. Priorities range from a low of 0 to a high of 31; 0 through 15 are timesharing priorities and 16 through 31 are real-time priorities. Compare with **job scheduling priority**.

batch execution queue

An execution queue that can accept only batch jobs.

batch job

A detached process that sequentially runs one or more command procedures. The user defines the list of command procedures when submitting the job to a batch queue.

batch mode

An execution mode in which you can execute a command procedure by submitting the procedure to a batch queue. When resources are available, the system creates a detached process to execute the commands in the procedure. Usually, processes running in batch mode execute at a lower process priority, to avoid competing with interactive users for system resources.

beginning-of-tape (BOT) marker

A piece of photoreflective tape that delimits the beginning of the writable area on a tape volume.

binding

On an InfoServer system, a function that creates a **virtual device unit** on a local OpenVMS system.

block

On Files–11 disks, the basic unit by which disk space is allocated (512 8-bit bytes). On magnetic tape, the size of a block is determined by the user.

boot block

Block 0 on a disk. It contains the location and size of the **bootstrap image**, which is used to boot the system. Certain processors, in order to boot, must read the boot block to obtain the location of the bootstrap image.

booting

Also called bootstrapping, the process of loading system software from the system disk into processor memory. You must install the operating system before you boot the system for the first time. See also **conversational boot** and **nonstop boot**.

bootstrap image

An image used to boot the system.

On VAX systems, the bootstrap image is VMB.EXE.

On AXP systems, the bootstrap image is APB.EXE.

bootstrapping

See **booting**.

bpi

Bits per inch; a measure used for characters of data on tape. Also called **density**.

caching

A performance enhancement in which the system stores information in memory; this includes information about a disk volume's free space, file identifications, quota file entries, and file headers.

capability

On VAX systems, software that makes the services of the vector processor available to system users.

circuit

In a network, a communications data path that connects **adjacent nodes**. A circuit is not a physical data path but, rather, a logical connection that operates over a physical connection (a line). All input and output (I/O) between nodes takes place over circuits.

cluster

On Files–11 media, a logical grouping of blocks; the basic unit by which disk space is allocated.

See also **VAXcluster system**, **VMScluster system**.

command procedure

A file containing DCL commands and, optionally, data used by those commands. When you execute a command procedure, the system reads the file and executes the commands it contains. This eliminates the need for you to enter each command separately. You can use command procedures to efficiently perform routine tasks. A command procedure can also be executed in **batch mode**.

command string

The complete specification of a command, including the command name, command qualifiers, parameters, and parameter qualifiers. Because a command can be continued on more than one line, the term is used to define the entire command.

Compact Disc Read-Only Memory (CD–ROM)

Computer discs similar to the CD–ROMs used for audio applications. The major difference is that CD–ROM computer disc players have a digital (rather than an audio) interface.

configuration database

In a network, each node has a configuration database that includes information about the node and other nodes with which it can communicate. The configuration database is made up of a **permanent database** and **volatile database**.

connection manager

In a VMScluster environment, the component that dynamically defines the VMScluster system and coordinates participation of computers in the cluster.

conversational boot

A booting operation in which you stop to perform special operations—for example, to change system parameter values—before booting. Contrast with **nonstop boot**.

Conversational boot operations are common in programming research and development environments where you must alter operating conditions for experimentation, testing, and debugging.

crash dump

When the operating system detects an unrecoverable error or an inconsistency within itself that causes the system to fail, it writes the contents of the error log buffers, processor registers, and memory into the **system dump file**.

crash history file

A file storing information about system crashes. Use the Crash Log Utility Extractor (CLUE) to display the contents of the crash history file to understand and resolve the issues responsible for crashes, and to obtain other useful data.

current accounting file

In a VMScluster environment, an **accounting file** for a particular node. By default, the current accounting file is SYS$MANAGER:ACCOUNTNG.DAT.

current values

With system parameters, the set of values that is stored in the default parameter file on disk and are used to boot the system. When the system boots, it reads the current parameter values into memory to create **active values**.

cylinder

On a disk, consists of all **tracks** at the same radius on all recording surfaces of the disk.

data area

One of two divisions of CD–ROM volume space; includes the remaining volume space, beginning with logical sector 16.

DECnet for OpenVMS

The name for the software and hardware products that allow various Digital operating systems to participate in a network. DECnet for OpenVMS allows a system to function as a node in a network.

default values

With system parameters, the set of values provided on your distribution kit and stored in the default list. These values allow you to boot any supported configuration.

density

A measurement, in bits per inch, used for characters of data on tape.

device control library

A text library that contains user-written modules consisting of text or escape sequences. See also **device control module**.

device control module

A user-written module in a **device control library**. Device control modules can be used for the following purposes:

- With programmable printers, to insert device-dependent escape sequences that set up a printer for selected print options such as point size, character set, and bold or italic print.
- With both programmable and non programmable printers, to insert text at specific points in the processing of a print job.

See also **page setup module**, **reset module**, and **setup module**.

device driver

A system component that controls I/O operations for a particular device type. For a device to function on a system, the device must be connected and the device driver must be loaded into memory.

disk

Physical media on which files reside.

disk quota

A method for maintaining and enforcing limits on the amount of disk space available to users on a public volume. See also **quota file**.

dynamic load leveling

A capability of a tightly coupled **multiprocessing system**, where the scheduler can assign a job to any processor on the basis of which processor is free to execute the job.

end node

In a network, a node that does not perform routing operations.

end-of-tape (EOT) marker

A piece of photoreflective tape that delimits the end of the writable area on a tape volume.

ERRFMT process

System process that periodically empties the error log buffers, transforms the descriptions of the errors into standard formats, and stores the formatted information in the **error log file** on the system disk.

error log file

The operating system automatically records device and CPU error messages in this file. The Error Log utility invokes the **Error Log Report Formatter (ERF)** to selectively report the contents of an error log file.

Error Log Report Formatter (ERF)

A system component invoked by the Error Log utility to selectively report the contents of the **error log file**.

Ethernet

A single shared network channel, with all nodes having equal access to the channel. Ethernet offers local and remote connections as one integral network.

event classes

Categories of security-relevant events. The system always audits several event classes.

executable image

An image that can be run in a process. It is linked with the /EXECUTABLE qualifier (or without the /SHAREABLE qualifier) of the Linker utility.

execution queue

A queue that accepts batch or print jobs for processing. Compare with **generic queue**.

executive

A set of programs in the operating system that control the running of routines that perform I/O, resource allocation, and program execution. See also **executive routines**.

executive mode

The second most privileged processor **access mode**. OpenVMS Record Management Services (RMS) and many system service procedures execute in executive mode.

executive routines

System routines that detect errors and events and write relevant information into error log buffers in memory. See also **executive**.

expiration date

The Files–11 On-Disk Structure uses the expiration date of a file to track the use of a file. The expiration date aids in the disposal of seldom-used files.

extent

On Files–11 volumes, contiguous blocks allocated to a particular file.

feedback

Information, continuously collected by the **executive**, about the amount of various resources the system uses to process its work load. When run in feedback mode, AUTOGEN analyzes this information and adjusts the values for any related system parameters.

field

In a UAF record, a portion of the record you modify with the Authorize utility. The values you assign to each field do the following:

- Identify the user
- Define the user's work environment
- Control use of system resources

file

On Files–11 media, an array of consecutive virtual blocks, numbered 1 to *n*, plus a set of attributes with values. A file is either a data file or a directory file. Directories can contain both data files and directory files.

file banner page

A **banner page** that separates files within a job; users can override the file banner page settings you set for a queue.

file header

On a Files–11 volume, describes a portion of a file on the volume. File headers contain information such as the **owner UIC**, **protection code**, creation date and time, and **access control list (ACL)**.

file operation

In the Backup utility, an operation that processes individual files or directories.

Files–11 On–Disk Structure

A logical structure given to information stored on a disk; it is a hierarchical organization of files, their data, and the directories needed to gain access to them.

Files–11 volume

A disk volume that uses Files–11 On-Disk Structure and is mounted on a device.

full backup

See **image backup**.

general timesharing service

A LAT service offering processing resources to users in the LAN. Contrast with **application service**.

generic batch queue

A generic queue that can direct jobs only to batch execution queues.

Generic batch queues are typically used in VMScluster environments to distribute the batch work load across several nodes.

generic output queue

A generic queue can direct jobs to any output execution queue. Generic output queues are typically used to distribute the output work load among several identical printers.

generic queue

A queue that holds batch or print jobs until they are transferred to an **execution queue** for processing.

A generic queue holds a job until an appropriate execution queue becomes available to initiate the job. The **queue manager** then requeues the job to the available execution queue.

group volume

A volume available to all the users in a group. Compare to **system volume**.

header labels

On magnetic tape, labels containing information such as the file name, creation date, and expiration date. When you create a file on magnetic tape, the magnetic tape file system writes header labels immediately preceding the data block. To access a file on magnetic tape by the file name, the file system searches the tape for the header label set that contains the specified file name.

header resident image

A **known image** for which the header of the image file remains permanently resident in memory, saving one disk I/O operation per file access.

home block

A block in a Files–11 volume that identifies it as a Files–11 volume. Usually, the home block is the next block after the **boot block** (block 0). If for some reason the home block cannot be read (is physically unusable), an alternative block is selected for use as the home block. This block provides specific information about the volume and default values for files on the volume.

identification record

A record of a **file header** that contains a summary of disk and volume characteristics.

image

A collection of procedures and data bound together by the Linker utility to form an executable program. Executable programs can be executed (or run) by a process. Usually, executable programs have the file type .EXE.

image backup

Also called a full backup. A Backup utility operation that saves a copy of all the files on a disk (or volume) to a special file called a **save set**. See also **image operation**.

image compare

A Backup utility operation that compares the contents of entire volumes.

image copy

A Backup utility operation that creates a new Files–11 On-Disk Structure on the output disk and copies an entire volume; the image backup is a logical duplicate of the contents of the disk.

image operation

A Backup utility operation that processes all files on the input disk.

image registry

A file associated with the Image Registry facility. To continue using a compatible application image that depends on a previous operating system version, you can register the image in the Image Registry.

image restore

A Backup utility operation that initializes the output disk and restores an entire volume.

incremental backup

A Backup utility operation that saves only those files that have been created or modified since the most recent backup that was performed using the /RECORD qualifier. (The /RECORD qualifier records the date and time that the files are backed up.)

incremental restore

A Backup utility operation that restores an incremental **save set**.

InfoServer system

An Ethernet-based, high-performance, **virtual device server**. The InfoServer system can serve physical device media and sets of logical disk blocks to client systems in a local area network (LAN). Systems running the appropriate client software can connect to virtual devices served by the InfoServer system and use them as though they are locally attached devices.

initialization file

In certain utilities, a file used each time you invoke the utility. In the initialization file, you can perform tasks such as defining keys and setting up your environment.

installation procedure

The procedure for installing the operating system for the first time. Also, a procedure for installing a layered product.

IRG (interrecord gap)

On magnetic tape, the interval of space between blocks.

job banner pages

banner pages that identify jobs; users cannot override job banner pages that you set for a queue. Compare with **file banner pages**.

job controller

The system process that creates a process to perform the tasks in a batch job.

job scheduling priority

A priority value that the system uses to schedule a batch or print jobs in a queue. Job scheduling priorities range from a low of 0 to a high of 255. Compare with **base process priority**.

kernel mode

The most privileged processor **access mode**. The operating system's most privileged services, such as I/O drivers and the pager, run in kernel mode. When in kernel mode, the processor has complete control of, and responsibility for, the system.

known file list

An internal data structure on which the system defines **known images**. Each entry in the known file list identifies the file name of the known image and the attributes with which it was installed.

known image

An image installed with the Install utility (INSTALL). When you install an image, the image is assigned attributes and becomes known to the system.

LASTport protocol

A specialized LAN transport protocol, implemented by the InfoServer software, that allows many clients to access InfoServer systems and perform reliable device read and write operations.

The LASTport/DISK protocol and LASTport/TAPE protocol are specialized disk and tape protocols that use the LASTport protocol.

See also **InfoServer system**.

LAT protocol

Protocol, implemented by the LAT software, that allows the operating system to offer resources, or LAT services that **terminal servers** can access.

LAT service announcements

Multicast messages sent by **LAT service nodes** and used to create a database of service nodes available.

LAT service node

A system that supports incoming LAT connections or a system that offers **LAT services**.

LAT services

Computing resources made available to users in the LAN through the LAT software. A LAT service can be a **general timesharing service** or an **application service**.

level 1 router

In a network, a node that performs routing operations within a single area. Compare with **level 2 router**.

level 2 router

In a network, a node that performs routing operations between areas and within its own area. Also called an **area router**. Compare with **level 1 router**.

license

Many software vendors provide software to their customers under an agreement called a license. Although the term *license* can have specific legal connotations, for the purpose of this manual a license refers to the authorization you have to use a product.

The License Management facility (LMF) lets you register, manage, and track software licenses on line. See also **Product Authorization Key (PAK)**.

line

In a network, a physical data path that connects **adjacent nodes**. A communications line connects your computer to the DECnet for OpenVMS network.

load address

The location in memory (specified in hexadecimal notation) to which the system loads the **bootstrap image**.

Local Area VAXcluster configuration

A VAXcluster configuration in which a single VAX computer serves as the management center of the cluster, plus one or more VAX computers that are connected to this hub.

local cluster

In the System Management utility (SYSMAN), the node from which you are executing SYSMAN.

local node

In a network, the node on which you are working.

In the System Management utility (SYSMAN), the node on which you execute SYSMAN.

Contrast with **remote node**.

logical block

Organizational unit of volume space. The logical block size cannot exceed the logical sector size.

logical block numbering

Begins with the first byte in the volume space and continues in a sequentially ascending order through the remainder of the volume space.

logical link

In a network, connects two processes and carries a stream of two-way communications traffic between the processes over a **circuit**. A single circuit established between two nodes can support many logical links concurrently.

logical queue

A special type of generic output queue that transfers print jobs to another output execution queue. You might use this kind of queue to temporarily redirect a queue when the device on which it runs is broken.

logical sector

Organizational unit of a volume; consists of one or more physical sectors. No more than one logical sector can begin in any physical sector.

Logical sectors are numbered in ascending order, with 0 assigned to the logical sector having the lowest physical address containing recorded data. Each logical sector includes a data field made up of 2048 or more bytes (the number of bytes always equals a power of 2).

login command procedure

A command procedure that executes each time a user logs in. Add commands to a login command procedure to execute commands when a user logs in, for example, to set up the user environment.

login (LGI) system parameters

System parameters that control login functions. The names of these system parameters begin with LGI.

loopback tests

In a network, a series of tests to help determine whether the network is operating properly.

lost file

A file that is not linked to a directory. When you delete a directory file (a file with the file type .DIR) without first deleting its subordinate files, the files referred to by that directory become lost files. Lost files are a nonproductive use of disk space and act as debits against a user's **disk quota**.

Magnetic Tape Ancillary Control Process (MTACP)

The internal software process of the operating system that interprets the logical format of standard labeled tape volumes.

maintenance release

A release of the operating system that is applied with an **update procedure**.

mandatory update

A software update that is required immediately after upgrading or installing the operating system.

mass storage control protocol (MSCP) server

In a VMScluster environment, the component that implements the MSCP protocol, which is used to communicate with a controller for DSA disks, such as RA-series disks. In conjunction with one or both of the disk class **device drivers** (DUDRIVER, DSDRIVER), the MSCP server implements this protocol on a computer, allowing the computer to function as a storage controller.

master file directory (MFD)

The file that contains the name of all user file directories on a disk.

mount verification

A recovery mechanism for disk and tape operations. If a device goes off line or is **write-locked** while mount verification is enabled, you can correct the problem and continue the operation.

multivolume file

A file that is continued on another volume when the data blocks of a file or related files do not physically fit on one volume (a reel of magnetic tape).

network

A means of connecting computers that allows them to share or transfer information or communications. A network includes two or more computers that are connected, and the hardware and software that makes those connections.

network proxy account

A user account that allows users on a **remote node** in a network to access data by way of a local account on your system. Proxy accounts are useful when you want to grant one or more users on a remote node access to specific files but you do not want to give them a private account on your system.

node

In a network, a computer system that is connected to another system in a network—by means of cables, telephone lines, microwave and satellite links, for example.

nonlocal cluster

In the System Management utility (SYSMAN), any cluster other than the one from which you are executing SYSMAN.

nonlocal environment

In the System Management utility (SYSMAN), your environment when you are not working on your local node or within your own cluster.

nonstop boot

The most common booting operation. You perform a nonstop boot if you do not want to stop to perform special operations—for example, to change system parameter values—before booting. Contrast with **conversational boot**.

object

In a network, a process to which a **logical link** connects. Some objects are DECnet programs—for example, the MAIL object; other objects are user-written programs.

For two programs to communicate over the network, the source program on the local node establishes a **logical link** with the object on the remote node.

OPCOM messages

Messages broadcast by the Operator Communication Manager (OPCOM). These messages are displayed on **operator terminals** and written to the **operator log file**. The messages might be general messages that you send, user requests, operator replies, or system events.

OPCOM process

The system process that manages Operator Communication Manager (OPCOM) operations.

operator log file

The Operator Communication Manager (OPCOM) records messages in this file. The file is named SYS$MANAGER:OPERATOR.LOG.

operator terminals

Terminals designated to display messages broadcast by the Operator Communication Manager (OPCOM). Usually, the console terminal (with the device name OPA0:) is the operator terminal. However, you can designate any user terminal as an operator terminal.

output execution queue

A queue that accepts jobs for processing by a **symbiont**. The **queue manager** sends the symbiont a list of files, which the user defines when submitting the job. An output symbiont transfers data from a disk to an output device. As the symbiont processes each file, it produces output for the device it controls, such as a printer or a terminal.

owner UIC

Used with **UIC-based protection**, usually the UIC of the person who created a file or volume.

page

A unit used for allocating and deallocating memory.

On VAX systems, a page is 512 bytes.

On AXP systems, a page can be 8 kilobytes (KB) (8192 bytes), 16 KB, 32 KB, or 64 KB. The initial set of AXP computers use a page size of 8192 bytes. Compare with **pagelet**.

page file

In a **paging** operation, the file to which the system writes paged portions of memory. Your distribution kit includes a page file named SYS$SYSTEM:PAGEFILE.SYS. If necessary, SYS$SYSTEM:PAGEFILE.SYS can be used in place of the system crash dump file.

pagelet

On AXP systems, a unit of memory in a 512-byte quantity. One AXP pagelet is the same size as one VAX page. Also, on an AXP 8KB computer, 16 AXP pagelets equal 1 AXP page.

page setup module

A **device control module** inserted at the beginning of each page of a print job.

paging

A memory management operation to efficiently use the physical memory allotted to a process by moving information between physical memory and files stored on disk. In paging, the system moves infrequently used portions of a process workspace out of physical memory to a file. Compare with **swapping**.

PAK

See **Product Authorization Key (PAK)**.

partition

A logical subset of a read/write disk. A single disk can be subdivided into several partitions, each of which of which can be used independently. The partitions appear to be whole disks.

permanent database

In a network, a permanent copy of the DECnet for OpenVMS **configuration database**. When you start the network, the permanent database provides the initial values for the **volatile database**. Changes remain after the network is shut down, but do not affect the current system.

permanently open image

A **known image** where directory information on the image file remains permanently resident in memory, eliminating the usual directory search required to locate a file.

physical dump

A **crash dump** containing the entire contents of physical memory to the **system dump file**. Compare with **selective dump**.

physical operation

In the Backup utility, an operation that copies, saves, restores, or compares an entire volume by logical blocks, ignoring any file structure.

physical sector

Division of a system or data area; smallest addressable unit on an ISO 9660 CD–ROM.

primary page and swap files

The default **page file** and **swap file** provided with your distribution kit. These files are named SYS$SYSTEM:PAGEFILE.SYS and SYS$SYSTEM:SWAPFILE.SYS. Contrast with **secondary page and swap files**.

primary processor

In a multiprocessing system, the processor that is either logically or physically attached to the console device and is the target of the console commands that bootstrap the multiprocessing system. The primary processor is responsible for starting other processors in the multiprocessing system. It also serves as the system timekeeper.

print forms

You can use print forms with output queues to determine certain page formatting attributes (such as margins and page length). In addition, the paper stock specified in a form determines whether a job is printed; if the stock of a job's form does not match the stock of the form mounted on the queue, the job is not printed

Digital supplies a default print form named DEFAULT. You can create additional forms if users need help formatting output, or if certain print jobs require special paper.

print job

An entry in an output queue that specifies a file or files to be printed on a printer. The user defines the file or files to be printed when submitting the job. When a printer is available, the **queue manager** sends the file to a **symbiont** for formatting and printing.

printer queue

A type of output execution queue that uses a **symbiont** to direct output to a printer. Compare with **server queue** and **terminal queue**.

priority

See **base process priority** or **job scheduling priority**.

private volume

A file-structured disk volume that contains only private files.

privileged image

A **known image** where increased **privileges** are temporarily assigned to any process running the image, permitting the process to exceed its user authorization file (UAF) privilege restrictions during execution of the image. In this way, users with normal privileges can run programs that require higher-than-normal privileges.

privileges

A means of restricting the functions users are authorized to perform on the system. System managers require privileges that are denied to most users.

process limits and quotas

User authorization file (UAF) parameters you can set for a user account to control the usage of system resources by processes in that account. (UAF parameters are different than system parameters.) You set values for process limits and quotas using the Authorize utility.

Product Authorization Key (PAK)

Information, typically on a piece of paper, provided for many Digital products. The data provided in the PAK allows you to register a software **license** in the license database on a system.

protected image

A **known image** that is a **shareable image** and contains protected code. Protected code is code that runs in **kernel mode** or **executive mode** but that can be called by a **user mode** image.

protection code

Used with **UIC-based protection**, indicates who is allowed access and for what purposes.

public volume

A Files–11 volume that any user on the system can access and that can contain both private and public files.

queue

Allows users to submit requests for printing or batch processing. The system prints users' print jobs or processes users' batch jobs as resources allow.

queue characteristics

Characteristics you can define and assign to a queue To control the batch or print jobs that execute on the queue.

queue database

A file or files that store information about queues and batch and print jobs.

queue manager

The system component that controls queue activity.

quota file

On Files–11 volumes, the file that records all users who are allowed to use a disk and that shows their current disk usage and their maximum disk allocation. A quota file, QUOTA.SYS, which is stored in directory [000000] with other system files, requires 1 block of disk storage for every 16 entries. See also **disk quotas**.

record blocking

On Files–11 volumes, the grouping of individual records into a block, thereby reducing wasted space.

remote node

In a network, a node that is accessible to the node you are working on (the local node) over the network.

In the System Management utility (SYSMAN), any node other than the one on which you are executing SYSMAN.

Contrast with **local node**.

reset module

A **device control module** inserted at the end of each print job. Use reset modules to reset a printer at the end of a job.

resident image

On AXP systems, a **known image** that improves the performance of a **shareable image**. With a resident image, portions of images that contain code are moved into system space, where they reside on a large single page, thus improving performance.

root volume

The first volume in a **volume set**. Each volume in the volume set is identified by a volume number relative to the root volume, which is always relative to volume 1.

router

In a network, a node that performs routing operations.

routing

In a network of more than two nodes, the process of directing a data message from a source node to a destination node (known as an **end node**). Both routers and end nodes can send messages to and receive messages from other nodes in the network.

save set

A special file used by the Backup utility. The Backup utility saves files to a save set and restores files from a save set. Installation and upgrade procedures restore product files from a save set to your system disk.

scalar

A single data item, having one value. Compare with **vector**.

secondary page and swap files

Additional **page files** and **swap files** that you might create for performance or disk space reasons. The system uses the space in the secondary files for paging and swapping in addition to the space in the **primary page and swap files**.

secondary processor

In a multiprocessing system, any processor that is not a **primary processor**.

sector

The smallest unit discernible to the Files–11 On-Disk structure. For most Files–11 disks, a sector is equivalent to a block (512 bytes).

On ISO 9660 volumes, a uniquely addressable unit; each sector on a CD–ROM comprises a sequence of 2048 8-bit bytes.

security audit log file

A clusterwide file that contains a record of security events on the system. Using the ANALYZE/AUDIT command, you can produce reports and summaries of security events from the security audit log file.

selective dump

A **crash dump** containing only those portions of memory most likely to be useful in a crash dump analysis. A selective dump is useful when sufficient disk space is not available to hold all physical memory. Compare with **physical dump**.

selective operation

A Backup utility operation that processes files or volumes selectively, according to criteria such as version number, file type, UIC, date and time of creation, expiration date, or modification date.

sequential organization

On magnetic tape media, the organization of data; that is, data is organized in the order in which it is written to the tape.

server queue

A type of output execution queue that uses a user-modified or user-written **symbiont** to process the files that belong to print jobs in the queue. Compare with **printer queue** and **terminal queue**.

setup module

A **device control module** inserted at the beginning of a file in a print job.

shareable image

An image linked with the /SHAREABLE qualifier of the Linker utility; it must subsequently be linked into an executable image to be used. Shareable images are sometimes referred to as **linkable images**.

shared image

A **known image** for which more than one user can access the read-only and non-copy-on-reference read/write sections of the image concurrently, so that only one copy of those sections ever needs to be in physical memory.

shared resource

In a VMScluster environment, a resource (such as a disk or a queue) that any node in the cluster can access. Data files, application programs, and printers are some items that can be accessed by users on a cluster with shared resources, without regard to the particular node on which the files or program or printer might physically reside.

site-independent startup command procedure

A command procedure that executes each time a system boots, and manages startup of a system. This file, named SYS$STARTUP:STARTUP.COM, is required on all systems, regardless of site-specific requirements. *Do not modify this file.* Compare with **site-specific startup command procedure**.

site-specific startup command procedure

A command procedure that executes each time a system boots. Unlike the **site-independent startup command procedure**, you can add commands to site-specific procedures to perform operations that vary from site to site.

sizing

The process of matching the allocation of system resources (memory and disk space) with the workload requirements of your site. Use the AUTOGEN command procedure to automatically size your system.

slicing

On AXP systems, a feature that lets the operating system split the contents of images and sort the pieces so that they can be placed with other pieces that have the same page protection in the same area of memory. Consequently, translation buffers on AXP systems are used more efficiently than if the loadable executive images or the shareable images were loaded in the traditional manner.

source disk

In the command procedures VMSINSTAL.COM or VMSKITBLD.COM, the disk from which you copy files. Compare with **target disk**.

spooled printer

A printer set up to write output to an intermediate storage device (such as a disk). Spool printers if your system runs applications that write or copy data directly to printers rather than submitting print jobs to a queue. In this way, printers remain available to other system users while the program is running.

startup database

A file that contains information used to start up system software. For example, the **site-independent startup command procedure** uses information in a startup database named STARTUP$STARTUP_VMS to start the operating system. It uses information in a startup database named STARTUP$STARTUP_LAYERED to start layered products.

swap file

In a **swapping** operation, the file to which the system writes swapped portions of memory. Your distribution kit includes a swap file named SYS$SYSTEM:SWAPFILE.SYS.

swapping

A memory management operation to efficiently use the physical memory allotted to an entire system by moving information between physical memory and files stored on disk. In swapping, the system moves the entire workspace of a less active process out of physical memory to a file. Compare with **paging**.

symbiont

Used with an output queue, a process for formatting of print jobs and sending them to a printer.

The standard print symbiont provided by the operating system is named PRTSMB and is designed to print files on basic output devices. The LAT print symbiont LATSYM is used to print files on output devices attached to a **terminal server**.

SYSGEN parameters

See **system parameters**.

system area

One of two divisions of CD–ROM volume space; includes logical sectors 0 through 15. Reserved for system use.

System Communications Services (SCS)

In a VMScluster environment, software that implements intercomputer communication, according to the Digital Systems Communications Architecture (SCA).

system disk

Disk on which operating system files are stored.

system dump file

The file into which the operating system writes the contents of the error log buffers, processor registers, and memory when it detects an unrecoverable error or an inconsistency within itself that causes the system to fail. See also **crash dump**.

system image

An image that does not run under the control of the operating system. It is intended for standalone operation only. The content and format of a system image differs from that of a **shareable image** and an **executable image**.

system image snapshot

A record of the system setup used with the Snapshot facility.

system messages

Messages returned by the system when you enter commands in DCL or in utilities. These messages help you understand the result of each command.

system parameters

Parameters for which you can set values to control how the system functions. Values of system parameters control a wide range of system functions including but not limited to memory management, process scheduling, and system security.

system volume

A volume available to all the users on a system. Compare to **group volume**.

systemwide logical name

A logical name that applies to the entire system. It is defined in the system logical name table and can be used by any process in a system.

tape mass storage control protocol (TMSCP) server

In a VMScluster environment, the component that implements the TMSCP protocol, which is used to communicate with a controller for local MSCP tapes, such as TU-series tapes. In conjunction with the tape class **device driver** (TUDRIVER), the TMSCP server implements this protocol on a processor, allowing the processor to function as a storage controller.

target disk

In VMSINSTAL.COM or VMSKITBLD.COM, the disk to which you move the system files. Compare with **source disk**.

terminal queue

A type of output execution queue that uses a **symbiont** to direct output to a terminal printer. Compare with **printer queue** and **server queue**.

terminal servers

Communication devices dedicated for connecting terminals, modems, or printers to a local area network (LAN) and to other systems within a LAN. See also **LAT protocol**.

track

On a disk, the collection of **sectors** (or blocks, on Files–11 volumes) at a single radius on one recording surface of the disk. It is accessible to a given read/write head position on the disk device.

trailer labels

On magnetic tape, labels similar to **header labels**, but written following the file.

trusted logical names

Logical names associated with **executive mode** or **kernel mode**.

tuning

The process of altering various system values to obtain the optimum *overall* performance possible from any given configuration and work load.

UAF

See **user authorization file (UAF)**.

UIC

See **user identification code (UIC)**.

UIC-based protection

A protection mechanism based on the **user identification code (UIC)** and applied to all protected objects. Compare with **access control list (ACL).**

update procedure

Procedure used if you have a previous version of the operating system and you want to make minor fixes to it. When you update the operating system, the update procedure replaces some system files.

upgrade procedure

If you are already running a standard version of the operating system, you can use the upgrade procedure to obtain a higher version.

user authorization file (UAF)

A file containing an entry for every user that you authorize to gain access to the system. Each entry identifies the user name, password, default account, UIC (user identification code), quotas, limits, and privileges assigned to individuals who use the system.

user identification code (UIC)

The pair of numbers assigned to users, files, and other system objects, that specify the type of access available to the owner, group, world and system. The UIC consists of a group number and a member number separated by a comma and enclosed within square brackets. Same as UIC. See also **account** and **UIC-based protection**.

user mode

The least privileged processor **access mode**. User processes and run-time library routines run in user mode.

utility program

A program supplied by Digital that performs a set of related operations. For example, the Backup utility (BACKUP) allows you to save and restore files.

VAXcluster satellite

In a Local Area VAXcluster configuration, a VAXcluster computer without a local system disk. A VAXcluster satellite uses disks and tapes locally connected to a **VAXcluster server**.

VAXcluster server

In a Local Area VAXcluster configuration, a VAXcluster node that uses the **mass storage control protocol (MSCP) server** and **tape mass storage control protocol (TMSCP) server** software to make its locally connected disks and tapes available to **VAXcluster satellites** over the local area network (LAN).

VAXcluster system

A loosely coupled configuration of two or more VAX computers and storage subsystems. A VAXcluster system appears as a single system to the user, even though it shares some or all of the system resources. When a group of VAX computers shares resources in a VAXcluster environment, the storage and computing resources of all the computers are combined, which can increase the processing power. See also **VMScluster system**.

VAXport drivers

In a VAXcluster environment, **device drivers** that control the communication paths between local and remote ports. (Examples are PADRIVER for the CI, PEDRIVER for the LAN, and PIDRIVER for the DSSI.)

vector

On VAX systems, a group of related **scalar** values, or elements, all of the same data type.

vector-capable systems

On VAX systems, those systems that comply with the VAX vector architecture.

vector consumer

On VAX systems, a process requiring the vector capability and having a vector context.

vector-present processor

On VAX systems, an integrated scalar-vector processor pair, included in a VAX vector processing system configuration.

virtual device server

Serves physical device media and sets of logical disk blocks to client systems in a local area network (LAN). Systems running the appropriate client software can connect to virtual devices as though they are locally attached devices. A virtual device server does not impose a file system on the virtual devices that it serves. See also **InfoServer system**.

virtual device unit

With an InfoServer system, a virtual device that represents the local OpenVMS context for a volume that resides on a remote server.

Virtual disk units have a device name in the DAD*n*: format. Virtual tape units have a device name in the MAD*n*: format.

See also **binding**, **InfoServer system**, and **virtual device server**.

VMScluster system

A loosely coupled configuration of two or more computers and storage subsystems, including at least one AXP computer. A VMScluster system appears as a single system to the user, even though it shares some or all of the system resources. When a group of computers shares resources in a VMScluster environment, the storage and computing resources of all the computers are combined, which can increase the processing power.

See also **VAXcluster system**.

volatile database

On a node in a network, a working copy of the DECnet for OpenVMS **configuration database** that reflects current network conditions. Contrast with **permanent database**.

volume

Disk or tape media that has been prepared for use by creating a new file structure on it and mounting it on a device.

volume set

A collection of disk volumes bound into a single entity by the DCL command MOUNT/BIND. To users, a volume set looks like a single, large volume.

Also, the volumes on which a set of multivolume files is recorded.

volume space

Set of all logical sectors on a volume containing information about the volume.

writable image

A **known image** for which a shared non-copy-on-reference writable section is removed from physical memory (for paging reasons or because no processes are referencing it), and it is written back to the image file.

write lock

A device becomes write-locked when a hardware or user error occurs while a disk or magnetic tape volume is mounted for a write operation. For example, if a disk is write-locked or a tape is missing a write ring, the hardware generates an error.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825) and press 2 for technical assistance.

Electronic Orders

If you wish to place an order through your account at the Electronic Store, dial 800-234-1998, using a modem set to 2400- or 9600-baud. You must be using a VT terminal or terminal emulator set at 8 bits, no parity. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825) and ask for an Electronic Store specialist.

Telephone and Direct Mail Orders

From	Call	Write
U.S.A.	DECdirect Phone: 800-DIGITAL (800-344-4825) FAX: (603) 884-5597	Digital Equipment Corporation P.O. Box CS2008 Nashua, NH 03061
Puerto Rico	Phone: (809) 781-0505 FAX: (809) 749-8377	Digital Equipment Caribbean, Inc. 3 Digital Plaza, 1st Street Suite 200 Metro Office Park San Juan, Puerto Rico 00920
Canada	Phone: 800-267-6215 FAX: (613) 592-1946	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: DECdirect Sales
International	————	Local Digital subsidiary or approved distributor
Internal Orders[1] (for software documentation)	DTN: 241-3023 (508) 874-3023	Software Supply Business (SSB) Digital Equipment Corporation 1 Digital Drive Westminster, MA 01473
Internal Orders (for hardware documentation)	DTN: 234-4325 (508) 351-4325 FAX: (508) 351-4467	Publishing & Circulation Services Digital Equipment Corporation NR02-2 444 Whitney Street Northboro, MA 01532

[1]Call to request an Internal Software Order Form (EN–01740–07).

Reader's Comments

OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems
AA–PV5NA–TK

Your comments and suggestions help us improve the quality of our publications.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (product works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

What I like best about this manual is ______________________________

What I like least about this manual is ______________________________

I found the following errors in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

For software manuals, please indicate which version of the software you are using: ____________

Name/Title ______________________ Dept. ____________
Company ______________________ Date ____________
Mailing Address ______________________
______________________ Phone ____________

--- Do Not Tear – Fold Here and Tape ---

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OpenVMS Documentation
110 SPIT BROOK ROAD ZKO3–4/U08
NASHUA, NH 03062–2642

--- Do Not Tear – Fold Here ---