VAX DSM
Installation and Management Guide

AA–HM24F–TE

September 1990

This document describes how to install and manage VAX DSM systems (Digital Standard MUMPS systems layered on the VMS operating system). It is intended for VAX DSM managers, operations personnel, and systems programmers with knowledge of the VMS operating system.

Revision/Update Information:	This is a revised document.
Operating System and Version:	VMS Version 5.2 and higher
Software Version:	VAX DSM Version 6.0

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, June 1986
Revised, September 1987
Revised, November 1988
Revised, November 1989
Revised, April 1989
Revised, September 1990

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013.

© Digital Equipment Corporation 1986, 1987, 1988, 1989, 1990

All Rights Reserved.
Printed in U.S.A.

The Reader's Comments form at the end of the hardcopy version of this document requests your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: DASL, DEC, DECnet, DECwindows, DSM–11, EDT, MicroVAX II, PDP–11, TK50, VAX, VAXcluster, VAX DOCUMENT, VAX DSM, VAX RMS, VMS, and the DIGITAL Logo.

MUMPS is a registered trademark of Massachusetts General Hospital.

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA*	**CANADA**	**INTERNATIONAL**
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

In Continental USA, Alaska, and Hawaii call 800–DIGITAL.

In Canada call 800-267-6215.

* Any order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Supply Business (SSB), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document is avaiable in both printed and CDROM versions.

This document was prepared with VAX DOCUMENT, Version 1.2.

Contents

Preface
Acknowledgment

Chapter 1 Installing VAX DSM

Required Operating System Components 1–1

VAX DSM Distribution Kit 1–1

Space and Time Requirements for Installing VAX DSM 1–4

Installation Overview 1–4

VAX DSM Installation 1–5
- Backing Up Your System 1–5
- Registering Your VAX DSM License 1–5
- Determining SYSGEN Requirements for Installation 1–6
- Installing VAX DSM Using VMSINSTAL 1–7

VAX DSM Shareable Images 1–14
- Installing VAX DSM as Shared Images 1–15
- Installing VAX DSM in a VAXcluster Environment 1–15

Postinstallation Activities 1–16
- Creating the VAX DSM Environment Manager's Account 1–16
- Running DSM$INIT.COM 1–18
- Configuring VAX DSM 1–24
- Editing the VMS System Files 1–26
  - System Startup 1–26
  - System Shutdown 1–27

Chapter 2 Managing VAX DSM

VAX DSM Environment 2–1
- Environment Directories, Files, and Protection 2–2
- VAX DSM Baseline Mode 2–5
- VAX DSM Configurations 2–5
- VAX DSM Accounts 2–6

Overview of VAX DSM Databases 2–7

Configuration Management (^CONMAN) 2–9
- Defining Configurations (^CONFIG) 2–9

Starting Up and Shutting Down Configurations 2–17
Starting a Configuration 2–17
Shutting Down a Configuration 2–19
Modifying Characteristics of a Configuration 2–20
Running Multiple VAX DSM Configurations 2–20

Initializing VAX DSM in a VAXcluster Environment 2–22
Directory Structure in a VAXcluster Environment 2–23
Defining VAXcluster Configurations 2–25

VAX DSM Detached Processes 2–25
Write Demon 2–26
Garbage Collector 2–26
Journal Process 2–27
Recovery Process 2–27
DDP Servers 2–27
DDP DECnet Link Servers 2–27

Chapter 3 Controlling Access to VAX DSM

Controlling User Access 3–1

VMS Access Control 3–4

Environment Access Control 3–4
Protecting Environment Files 3–5
Maintaining Environment Security 3–6

Configuration Access Control 3–7
VAX DSM Access Modes 3–7
Manager Mode Access 3–8
Programmer Mode Access 3–9
Application Mode Access 3–10

Global Access Control 3–10

Establishing VMS Accounts for Programmers and Application Users . . 3–11

Configuration Access Control Utilities 3–12
Granting Access Interactively (^ACL) 3–12
Granting Access Noninteractively (^ACLCALL) 3–15

Creating Dedicated Application Accounts 3–16
Setting Up a Captive Account 3–17
Creating a Captive Command Procedure 3–17
Setting Up VAX DSM Tied Routine and Port Tables 3–18

Controlling Access to the Callable Routines Interface 3–20

Using the Login Control Utility (^LOGIN) 3–21

Chapter 4 Tuning VMS for Use with VAX DSM

Overview 4–1

Process Quotas for VAX DSM Users 4–1

SYSGEN Parameters for VAX DSM Configurations 4–3
Using the ^MEMORY Utility 4–6
Modifying VMS System Parameters 4–10
Page and Swap File Size 4–11

Configuring VAX DSM Shared Memory 4–12
Configuring Nonpaged Memory 4–14

Chapter 5 Optimizing VAX DSM

Optimizing VAX DSM Applications 5–1
Using Mapped Routines 5–2
Mapped Routine Utilities (^RMAP) 5–3
Determining Memory Requirements of Mapped Routine Sections . . 5–4
Creating and Mapping Routine Sets 5–5
Executing Mapped Routines 5–9
Disabling Individual Mapped Routines 5–10
Considerations when Using Mapped Routines 5–12
Global Statistics (^GLSSTA) 5–13
Performance Statistics (^RTH) 5–18
Using the Performance Statistics Utility 5–18
Database Event Counts and Derivative Ratios Histogram . . 5–21
Individual Routine CPU Time as a Percentage of Total Routine CPU Time Histogram 5–26
Individual Routine Loads as Percentage of Total Histogram . 5–28
Global References Per Second Histogram 5–30
Distributed Data Processing Activity Histogram 5–32
Optimizing VAX DSM Based on Performance Statistics 5–34

Chapter 6 VAX DSM System Utilities

Overview of VAX DSM Utilities 6–1
Online Help 6–2
Utility Conventions 6–2
Running the VAX DSM System Utilities 6–4
Configuration Management Utilities (^CONMAN) 6–5
Displaying Configurations (DISPLAY^CONFIG) 6–5
Copying Configurations (COPY^CONFIG) 6–7
Modifying Configurations (MODIFY^CONFIG) 6–7
Deleting Configurations (DELETE^CONFIG) 6–8
Database Set Management Utilities (^DBSMAN) 6–8
After-Image Journaling Utilities (^JOURNAL) 6–8
Before-Image Journaling Utilities 6–9
Database Set Utilities 6–10
Volume Set Utilities (^VOLUTL) 6–10
Disk Block Tally (^DBT) 6–11
Distributed Data Processing Utilities (^DDP) 6–11
Mapped Routine Utilities (^RMAP) 6–12
Option File Management Utilities (^OPTMAN) 6–12
Performance Utilities 6–13
System Management Utilities (^SYSMAN) 6–14
Force Exit (^FORCEX) 6–15

Lock Table Display (^LOCKTAB) 6–15
Shutdown (^SHUTDWN) 6–15

Appendix A Installing VAX DSM Software in an Alternate Root

Using VAX DSM from an Alternate System Root A–1

Installing VAX DSM in an Alternate Root A–3
Running the DSM$ALTROOT_INSTALL.COM Procedure A–5
Installing VAX DSM in an Alternate Root in a VAXcluster Environment A–9
Shutting Down VAX DSM Configurations that Use Alternate Root Software A–9

Appendix B Using the Configuration Option File Utility (^OPTMAN)

Using the Configuration Option File B–1

Configuration Option File Utilities (^OPTMAN) B–2
Creating an Option File (^OPTBLD) B–2
Converting the Option File (^OPTCON) B–3

Starting Up a Configuration B–4

Configuration Option File Keywords B–4

Index

Examples

1–1 VMSINSTAL Console Output 1–11
1–2 Running the DSM$INIT Command Procedure 1–18
2–1 Running the ^CONFIG Utility 2–10
5–1 ^GLSSTA Report 5–14
B–1 Sample Configuration Option File B–21

Figures

1–1 VAX DSM Image Relationships 1–14
2–1 Directory Hierarchy in a Single-Node System 2–4
2–2 Directory Hierarchy in a VAXcluster Environment 2–23
3–1 Access Control Relationships 3–3
5–1 Database Event Counts and Derivative Ratios Histogram 5–22
5–2 Individual Routine CPU Time as a Percentage of Total Routine CPU Time Histogram 5–27

5–3 Individual Routine Loads as a Percentage of Total Routines Loads Histogram 5–29
5–4 Global References Per Second Histogram 5–31
5–5 Distributed Data Processing Activity Histogram 5–33
A–1 System Root SYS0 Directory Structure A–1
A–2 Alternate Root Directory Structure A–2

Tables

1–1 Space Requirements for Installing VAX DSM 1–4
3–1 VMS System Privileges Needed by VAX DSM Programmers 3–12
5–1 Routine Mapping Utilities 5–3
B–1 Definition Keywords B–6
B–2 Data Keywords B–10

Preface

Intended Audience

This manual addresses the VAX DSM™ manager and operator who has the responsibility for installing the VAX DSM system software, establishing VAX DSM user accounts, and performing other privileged system operations. This person should have extensive knowledge of the VMS™ operating system.

See the *VAX DSM Database Operations Guide* for information about managing volume sets and globals, maintaining the integrity of your database, using journaling, running VAX DSM in a VAXcluster™ environment, and using transaction processing.

All readers of this document should be familiar with the topics covered in the *Introduction to VMS*. The *VMS Software Product Description*, the *VMS Glossary*, and the *VMS Master Index* contain additional introductory information about the VMS operating system and describe where the information is located. All VAX DSM users should be familiar with the information in these publications before reading this document.

The *Introduction to VMS System Management* and the *Guide to Setting Up a VMS System* provide information for those who have the overall responsibility for controlling the operations of a VMS installation. If you are responsible for installing the VAX DSM system software and establishing VAX DSM user accounts, you should be familiar with the information in those manuals before reading this document.

Manual Objectives

This manual describes how to install the VAX DSM system software and how to operate and manage VAX DSM in a multiuser environment. It also describes how the VAX DSM global database is stored.

This manual does not describe the language elements or syntax of the VAX DSM language. See the *VAX DSM Language Reference Manual* for a complete description of the VAX DSM language elements and syntax. See the *VAX DSM Programmer's Guide* for information on programming using VAX DSM.

The *VAX DSM Installation and Management Guide* begins with installation and management information, followed by conceptual and detailed information that helps the system manager understand how VAX DSM functions. This manual is structured as follows:

- Chapter 1 provides instructions for installing a new VAX DSM system.
- Chapter 2 contains tips for managing a VAX DSM system, including instructions for defining a VAX DSM environment and for creating configurations.
- Chapter 3 describes how to control access to VAX DSM and how to create application and programmer accounts.
- Chapter 4 explains how to tune VMS for use with VAX DSM.
- Chapter 5 provides information on optimizing your VAX DSM system.
- Chapter 6 contains information on System Utilities that perform system management functions.
- Appendix A provides instructions on how to install VAX DSM in an alternate root.
- Appendix B describes how to use the Configuration Option File Utility (^OPTMAN) to define configurations.

Related Documents

This manual refers to the following documents from the VMS documentation set that contain supplemental information relevant to VAX DSM programming and system operations:

- *Guide to DECnet-VAX Networking*
- *Guide to Setting Up a VMS System*
- *Guide to VMS File Applications*
- *Introduction to VMS*
- *Introduction to VMS System Management*
- *VMS Authorize Utility Manual*
- *VMS Backup Utility Manual*
- *VMS DCL Dictionary*
- *VMS Glossary*
- *VMS Guide to System Security*
- *VMS Install Utility Manual*
- *VMS License Management Utility Manual*
- *VMS Master Index*
- *VMS System Generation Utility Manual*
- *VMS System Messages and Recovery Procedures: Part I and II*
- *VMS SYSMAN Utility Manual*
- *VMS VAXcluster Manual*

Conventions Used in This Document

This manual uses the following documentation conventions and symbols.

Convention	Meaning
Boldface	Emphasizes important information. Indicates user input for online documentation.
Italics	Introduces new terms. Indicates the title of a manual.
RED	Indicates user input for hardcopy documentation.
‘‘ ’’	Indicates a section title.
Ctrl/x	Indicates that you press the Ctrl key on the terminal keyboard while you simultaneously press some other key (represented here by x).
Return	Indicates that you press the Return key on the terminal keyboard.
SP	Indicates that a space must separate components of a command or command line.
Tab	Indicates that you press the Tab key.
[item]	Indicates that the enclosed item is optional. (However, square brackets are not optional in the syntax of a directory name in a VMS file specification.)
{item}	Indicates that the enclosed item is optional.
filespec	Indicates a VMS file specification.
. . .	Indicates that additional command parameters can be added to the command line.
. . .	Indicates a break between two illustrated lines of user input and that all user input is not shown.

Acknowledgment

Digital Standard MUMPS™ (DSM) is an extension of the American National Standards Institute (ANSI) Standard Specification for the Massachusetts General Hospital Utility Multi-Programming System (MUMPS). MUMPS was originally developed at the Laboratory of Computer Science at Massachusetts General Hospital and was supported by grant HS00240 from the National Center for Health Services Research and Development.

Chapter 1
Installing VAX DSM

This chapter provides information on the VAX DSM distribution kit and the software installation procedure which is used by sites that have never previously installed VAX DSM. Read the information in this chapter before you do a **new** installation of VAX DSM Version 6.0. **This chapter does not describe procedures used to upgrade previous versions of VAX DSM**. If you are upgrading from VAX DSM Version 5.0, 5.1, or 5.2, follow the instructions in Chapter 2 of the *VAX DSM Version 6.0 Release Notes*.

For more information about the VMS operating system, see the *Introduction to VMS System Management* and the *Guide to Setting Up a VMS System*.

Required Operating System Components

You can install VAX DSM Version 6.0 on systems running VMS Version 5.2 or higher. For information on VMS Tailoring Classes, please refer to the Software Support Addendum (SSA).

VAX DSM Distribution Kit

The VAX DSM distribution kit consists of software and documentation. Digital distributes the VAX DSM software on the following media:

- Magnetic Tape
- TK50™ Tape Cartridge
- CDROM

The VAX DSM software consists of the following:

- DSM interpreter and database handler image
- DSM Distributed Data Processing (DDP) driver
- Command procedures for managing a DSM application environment
- DSM library utility routines and globals
- Components needed to build user-defined functions

In addition to this manual, the documentation for VAX DSM Version 6.0 consists of:

- *Introduction to DSM*

  This document introduces the common syntax and language elements of DSM-11™ (Digital Standard MUMPS on the PDP-11™ computer) and VAX DSM (Digital Standard MUMPS layered on the VMS operating system).

- *VAX DSM Callable Routines Reference Manual*

  This document describes the callable interface, which allows routines written in software languages that run layered on the VMS operating system to access the DSM database and interpreter.

- *VAX DSM Database Operations Guide*

  This document describes how to maintain the integrity and reliability of the VAX DSM database. It also describes transaction processing.

- *VAX DSM Language Pocket Reference*

  This document summarizes VAX DSM language elements, the DSM command syntax, and I/O options.

- *VAX DSM Language Reference Manual*

  This document describes the syntax and elements of the VAX DSM language.

- *VAX DSM Programmer's Guide*

  This document describes how to use the programming capabilities of VAX DSM.

- *VAX DSM Version 6.0 Master Index*

  This document contains index entries for manuals in the VAX DSM documentation set (excluding the DASL™ manuals).

- *VAX DSM Version 6.0 Release Notes*

  This document contains new features, enhancements, and upgrade information for Version 6.0 of VAX DSM.

- *DASL Handbook*

  This document, for first-time users of DASL applications, describes how to use applications that were created with the DASL software. It also describes how to use the application environment options such as the Report Directory, the Message Center, and the Query Database Menu.

- *DASL Management Guide*

  This document provides information on DASL system management tasks such as setting up your terminal, initializing the DASL software, and performing database conversions.

- *DASL Pocket Reference*

  This document provides a summary of DASL commands and input conventions.

- *DASL Programmer's Guide*

  This document introduces the DSM Application Software Library (DASL) and explains how to create a simple application using DASL software.

- *DASL Reference Manual*

  This document describes the six major modules of the DASL software and provides related reference material, including the syntax and elements of DASL commands.

- *DASL Version 6.0 Master Index*

  This document contains index entries for manuals in the DASL documentation set.

The VAX DSM and DASL documentation sets are available on the VMS Online Documentation Library Compact Disk. These documentation files can be used with the VMS DECwindows™ Bookreader. The Bookreader is an online documentation viewer that is provided with the VMS DECwindows software.

Space and Time Requirements for Installing VAX DSM

To install and use VAX DSM, you must have enough space on your *system disk*. Table 1–1 shows the amount of space required for installing VAX DSM.

Table 1–1 Space Requirements for Installing VAX DSM

	With DASL	Without DASL
Peak Installation Utilization	30,000 blocks	18,500 blocks
Net Utilization	27,000 blocks	15,500 blocks

Actual disk space utilization depends on the requirements of your particular system environment.

On a MicroVAX II™ system, it takes approximately 15 minutes to install VAX DSM.

Installation Overview

This section provides an outline of how to install, initialize, and start up VAX DSM Version 6.0. The installation and startup procedure describes how to set up a new installation and assumes that there is no previous release of VAX DSM installed on your VMS system.

> **Note: You must be running VMS Version 5.2 or higher to install VAX DSM Version 6.0.**

The remainder of this chapter describes the following procedures in detail:

- VAX DSM software kit installation activities
  - — Back up your system.
  - — Register your VAX DSM license.
  - — Install the VAX DSM distribution kit onto your VMS system using the SYS$UPDATE:VMSINSTAL.COM procedure.
  - — If you do not run the IVP during VMSINSTAL, install the VAX DSM images as shared images using the SYS$STARTUP:DSM$INSTALL.COM procedure.

- Postinstallation activities
  - — Use the SYS$MANAGER:DSM$INIT.COM procedure to create a VAX DSM environment manager's account for use with VAX DSM.
  - — Edit the VMS system files to provide for automatic startup and shutdown of VAX DSM configurations when your system is rebooted.
  - — Log in to the VAX DSM environment manager's account and run the SYS$MANAGER:DSM$CONFIGURE.COM procedure to initialize VAX DSM database volume sets and configurations.

VAX DSM Installation

This section provides detailed instructions for installing the VAX DSM Version 6.0 software kit on your VMS system.

Backing Up Your System

Digital recommends that you do a system disk backup before installing any software on the operating system.

Use the backup procedures that have been established at your site. For details on performing a system disk backup, see the *VMS Backup Utility Manual* in the VMS documentation set.

Registering Your VAX DSM License

To run VAX DSM Version 6.0, you must register your software license by using the VMS License Management Facility (LMF). The license registration information you need is contained in the Product Authorization Key (PAK) that is shipped with VAX DSM. The PAK is a paper certificate that includes the data needed to authorize access to VAX DSM on your system.

The VMSINSTAL installation procedure asks if you have *registered* and *loaded* the DSM license. The VAX DSM kit includes an Installation Verification Procedure (IVP) that verifies if VAX DSM is installed correctly. You must register and load your DSM license before you start the installation in order to run the IVP during the installation. If you choose not to register and load your license, you can install DSM, but the IVP does not run.

To **register** a license, log in to the VMS system manager's account. Do one of the following to register the VAX DSM license:

- Invoke the SYS$UPDATE:VMSLICENSE.COM procedure. When it prompts you for information, enter the data from your PAK.
- Use the DCL LICENSE REGISTER command with the appropriate qualifiers that correspond to information on the PAK.

After you register the license, you must load the license to activate it. Enter the following to **load** the license:

```
$ LICENSE LOAD DSM Return
```

See the *VMS License Management Utility Manual* for additional information on using LMF, or type HELP LICENSE at the DCL prompt.

Determining SYSGEN Requirements for Installation

During the VAX DSM installation procedure, system resources are checked to ensure that sufficient free GBLPAGES and GBLSECTIONS exist in order to install the DSM images as *shared* image files. If system resources are not sufficient, the installation procedure terminates and notifies you.

To successfully install VAX DSM, you must have the following system resources available:

- GBLPAGES

  You need at least 550 global page table entries available.

- GBLSECTIONS

  You need at least 14 global section descriptors available.

You can use the F$GETSYI lexical function to display the number of free global pages and free global sections on your system. Enter the following DCL command line:

```
$ WRITE SYS$OUTPUT F$GETSYI("FREE_GBLPAGES") Return
8858
$ WRITE SYS$OUTPUT F$GETSYI("FREE_GBLSECTS") Return
68
```

The F$GETSYI lexical function displays the number of free global pages and free global sections that are available on your system.

If the installation procedure fails because these SYSGEN values are too low, you have to modify the system parameters, reboot your VMS system, and restart the installation procedure.

When adjusting system parameters, add at least 550 to the current value of GBLPAGES and at least 14 to the current value of GBLSECTIONS. To do this, examine the current system values using the System Generation Utility (SYSGEN):

```
$ RUN SYS$SYSTEM:SYSGEN Return
SYSGEN>  SHOW GBLSECTIONS Return

Parameter Name       Current   Default   Minimum   Maximum Unit Dynamic
--------------       -------   -------   -------   ------- ---- -------
GBLSECTIONS              250       250        20      4095 Sections

SYSGEN> SHOW GBLPAGES Return
Parameter Name       Current   Default   Minimum   Maximum Unit Dynamic
--------------       -------   -------   -------   ------- ---- -------
GBLPAGES               20000     10000       512        -1 Pages

SYSGEN>EXIT Return
```

Then, add the minimum values required by VAX DSM to the current values and place the new values in SYS$SYSTEM:MODPARAMS.DAT. For example, the current values from the previous example are adjusted in MODPARAMS.DAT using the following commands:

```
GBLSECTIONS = 250+14 ! 14 additional sections required by VAX DSM
GBLPAGES = 20000+550 ! 550 additional pages required by VAX DSM
```

Next you need to run the SYS$UPDATE:AUTOGEN.COM procedure to update the SYSGEN parameters. Then, reboot your system as follows:

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

AUTOGEN makes the necessary SYSGEN modifications and reboots your system. When the system reboots, you can rerun the software kit installation described in the next section.

Note that when you create new VAX DSM configurations, you may also need additional GBLPAGES and GBLSECTIONS. See Chapter 4 for more information on determining appropriate values for these and other SYSGEN parameters.

Installing VAX DSM Using VMSINSTAL

The following is an example VMSINSTAL procedure used to install VAX DSM Version 6.0:

1. To begin the installation procedure, log in to your VMS system account and enter the following command line:

   ```
   $ @SYS$UPDATE:VMSINSTAL Return
   ```

2. VMSINSTAL displays the following on the terminal:

   ```
              VMS Software Product Installation Procedure V5.2

   It is 16-OCT-1990 at 09:23.

   Enter a question mark (?) at any time for help.
   ```

Note: You can print a copy of the release notes using the OPTIONS N parameter when you enter the VMSINSTAL command line. For example:

```
$ @SYS$UPDATE:VMSINSTAL OPTIONS N Return
```

3. VMSINSTAL then asks the following questions:

```
* Are you satisfied with the backup of your system disk [YES]? Return
* Where will the distribution volumes be mounted: MUA0: Return
```

This sample installation procedure uses magnetic tape device MUA0. Enter the appropriate device mnemonic for your distribution medium at the prompt, and follow the procedures for mounting your distribution medium.

4. VMSINSTAL continues by displaying a prompt asking for the products you want to install from the first distribution volume set.

Type DSM after the following prompt and press Return:

```
* Products: DSM Return
```

If you did not specify OPTIONS N on the DCL command line, you can print a copy of the release notes by typing N after the Options prompt.

```
* Enter installation options you wish to use (none): N Return
```

5. The following prompts tell you to mount the first volume of the set and ask when you are ready. Type YES after the prompt and press Return. VMSINSTAL displays a message when the volume set is mounted:

```
Please mount the first volume of the set on MUA0:
*Are you ready? YES Return
%MOUNT-I-MOUNTED, DSM mounted on MUA0:
```

6. VMSINSTAL displays the following on the terminal:

```
The following products will be processed:

  DSM V6.0

        Beginning installation of DSM V6.0 at 09:24

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
```

If you typed an N after the prompt for installation options, VMSINSTAL displays the following:

```
Release Notes Options:

    1.  Display release notes
    2.  Print release notes
    3.  Both 1 and 2
    4.  Copy release notes to SYS$HELP
    5.  Do not display, print or copy release notes

* Select option [2]: Return
* Queue name [SYS$PRINT]: Return
Job DSM060 (queue SYS$PRINT, entry 25) started on SYS$PRINT
* Do you want to continue the installation? Y Return
```

Select the appropriate responses.

7. VMSINSTAL displays the following:

```
%VMSINSTAL-I-RELMOVED, The products release notes have been
successfully moved to SYS$HELP.
```

8. The following question asks if you have an authorization key registered and loaded. If you have an authorization key loaded, answer Y. If you do not have a key registered, answer N. If you answer N, you cannot run the IVP procedure.

```
        Product:       DSM
        Producer:      DEC
        Version:       6.0
        Release Date:  15-JAN-1990

*Does this product have an authorization key registered and loaded?Y Return
```

9. The VAX DSM kit includes an Installation Verification Procedure (IVP) called SYS$TEST:DSM$IVP.COM. VMSINSTAL optionally runs the IVP to verify that VAX DSM has been installed correctly. You can also run the IVP after running VMSINSTAL by entering @SYS$TEST:DSM$IVP at the DCL prompt. If you do not have a PAK loaded, VMSINSTAL will not run the IVP during the installation.

```
* Do you want to run the IVP after the installation [YES]? Return
```

10. VMSINSTAL asks if you want to install the DSM Application Software Library (DASL software) files with the following prompt:

```
* Do you want to include DASL files [YES]? Return
```

    The default response to the prompt is YES. Press Return if you want to include DASL files; type N if you do not want to include DASL files.

11. VMSINSTAL continues the VAX DSM Version 6.0 installation procedure by displaying the following prompt:

```
*Do you want to purge files replaced by this installation[YES]? Return
```

    The default response to this prompt is YES. Answer YES in response to this prompt to purge all previous VAX DSM files that VMSINSTAL replaces with new VAX DSM files. Allowing the previous VAX DSM files to remain on disk takes up valuable disk space that you may need.

12. VMSINSTAL prints the following informational message:

```
    No further questions will be asked during this installation
```

13. VMSINSTAL continues the installation procedure by copying the VAX DSM Version 6.0 images, utilities, globals, and supporting DCL command procedures. If you answer YES to installing the DASL software, VMSINSTAL creates DASL files in SYS$LIBRARY.

    VMSINSTAL displays informational messages on your terminal during each phase of this part of the installation procedure.

```
%VMSINSTAL-I-RESTORE, Restoring product save set B ...

The following files will be created or updated
.
.
.
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their
 target directories...
```

14. VMSINSTAL moves all of the VAX DSM Version 6.0 files to their appropriate directories and then executes the VAX DSM IVP.

    If the installation is successful, VMSINSTAL displays the following informational messages:

```
      Calling VAX DSM Installation Verification Procedure

      VAX DSM V6.0
         SYSTEM
       [Baseline]

      VAX DSM V6.0 IVP Successful

%DSM-I-HALT command executed

VAX DSM installation is complete.  You may enter the following
command line to initialize a manager account for use with VAX DSM:

               @SYS$MANAGER:DSM$INIT

      Installation of DSM V6.0 completed at 9:37
```

    If the installation fails for any reason, VMSINSTAL displays an appropriate informational message. See *VMS System Messages and Recovery Procedures: Part I and Part II* for information on installation failure messages. Take action based on the informational message.

15. VMSINSTAL prompts for additional software products to install:

```
* Products : EXIT Return

        VMSINSTAL procedure done at 09:37
```

    Enter another product to install from the next distribution volume set, or enter a ? for help. To exit the routine, press Ctrl/z, or type EXIT and press Return.

The VAX DSM software kit installation procedure is now complete. During the installation process the DSM060.RELEASE_NOTES are moved to the SYS$HELP directory. You can type or print these files to obtain the latest information about VAX DSM Version 6.0.

VMSINSTAL Procedure

Example 1–1 shows the console output of the VMSINSTAL procedure.

Example 1–1 VMSINSTAL Console Output

```
$ SET DEF SYS$UPDATE
$ @VMSINSTAL

It is 16-FEB-1990 at 09:23.

Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]?
* Where will the distribution volumes be mounted: MUA0:

Enter the products to be processed from the first distribution volume set.
* Products: DSM
* Enter installation options you wish to use (none):

The following products will be processed:

  DSM V6.0


  Beginning installation of DSM V6.0 at 09:24

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED , The product's release notes have been successfully
 moved to SYS$HELP.


            Product:      DSM
            Producer:     DEC
            Version:      6.0
            Release Date: 15-JAN-1990

* Does this product have an authorization key registered and loaded? Y
* Do you want to run the IVP after the installation [YES]?
* Do you want to include DASL files [YES]?
* Do you want to purge files replaced by this installation [YES]?

  No further questions will be asked during this installation

%VMSINSTAL-I-RESTORE, Restoring product save set B ...

%DSM-I-LINK, linking shareable image DSM$SHARE.EXE
%DSM-I-STB, creating DSM$SHARE.STB
%DSM-I-LINK, linking main image DSM.EXE
%DSM-I-LINK, linking DDP Driver image DSMDDPDRV.EXE
%LINK-W-USRTFR, image DISK$USER:[SYS0.SYSUPD.DSM060]DSMDDPDRV.EXE;1 has
no user transfer address

  Note: The %LINK-W-USRTFR Linker warning message above is expected
        while linking DSMDDPDRV.EXE

The following files will be created or updated

directory [SYSEXE]

  DSM.EXE               -  VAX DSM main image
  DSMDDPDRV.EXE         -  VAX DSM DDP driver
```

Example 1–1 Cont'd on next page

Example 1–1 (Cont.) VMSINSTAL Console Output

```
directory [SYSLIB]

 DSM$SHARE.EXE          -  VAX DSM shareable image
 DSM$SHARE.STB          -  Symbol table for DSM$SHARE.EXE
 DSM$MT_EBCASC.EXE      -  VAX DSM EBCDIC/ASCII translation tables
 DSM$MT_EBCASC.MAR      -  Source file of DSM EBCDIC/ASCII conversion table
 DSM$ZCALL.EXE          -  VAX DSM ZCALL package table
 DSM$ZCMACRO.MLB        -  Macro library for user written ZCALL packages
 DSM$CALLDEF.ADA        -  Definition file for callable routines
 DSM$CALLDEF.BAS        -  Definition file for callable routines
 DSM$CALLDEF.FOR        -  Definition file for callable routines
 DSM$CALLDEF.H          -  Definition file for callable routines
 DSM$CALLDEF.MAR        -  Definition file for callable routines
 DSM$CALLDEF.PAS        -  Definition file for callable routines
 DSM$CALLDEF.PLI        -  Definition file for callable routines
 DSM$CALLDEF.R32        -  Definition file for callable routines
 DSM$CALL.OPT           -  Linker option file for callable routines
 MANAGER.DSM            -  DSM Management routines
 UTILITIES.DSM          -  DSM Library routines
 DSMVOL1.GLS            -  DSM primary volume set
 MENU.GBL               -  Library utility menu global
 HELP.GBL               -  DSM HELP global
 ED.GBL                 -  DSM editor global
 EDI.GBL                -  DSM editor EDI global
 EDIHELP.GBL            -  DSM editor EDI HELP global
 EDT.GBL                -  DSM EDT editor global
 TPU.GBL                -  DSM TPU editor global
 Q.GBL                  -  DSM utility global for query generation
 REPLACE.GBL            -  DSM utility global for recompiling user routines
 RESTORE.GBL            -  DSM utility global for restoring DSM routines
 LIBREPL.GBL            -  DSM utility global for recompiling library routines
 DSM$GLOBAL.FDL         -  FDL file to CONVERT/FDL for RMS globals

directory [SYSMGR]

 DSM$INIT.COM           -  Command procedure to initialize a DSM account
 DSM$CONFIGURE.COM      -  Command procedure to create a DSM configuration
 DSM$UPGRADE.COM        -  Command procedure to upgrade a DSM configuration
 DSM$VOLUPG.COM         -  Command procedure to upgrade a volume set
 DSM$SHUTDOWN.COM       -  Command procedure to shut down a DSM configuration
 DSM$MSG_BUILD.COM      -  Command procedure to create custom error messages
 DSM$MT_BUILD.COM       -  Command procedure to build EBCDIC/ASCII tables
 DSM$LNKINI.COM         -  Command procedure to start a DECnet DDP link server

directory [SYSMSG]

 DSM$MESSAGE.MSG        -  DSM source file of error messages
 DSM$MESSAGE.EXE        -  DSM message file image
 DSM$MSGPTR.OBJ         -  DSM message pointer object file

directory [SYS$STARTUP]

 DSM$STARTUP.COM        -  Command procedure to start a DSM environment
 DSM$INSTALL.COM        -  Command file to install VAX DSM at VMS boot time

directory [SYSHLP]

 HELPLIB.LIB           - Add/update VAX DSM help text
 DSM060.RELEASE_NOTES  - VAX DSM Version 6.0 Release Notes
```

Example 1–1 Cont'd on next page

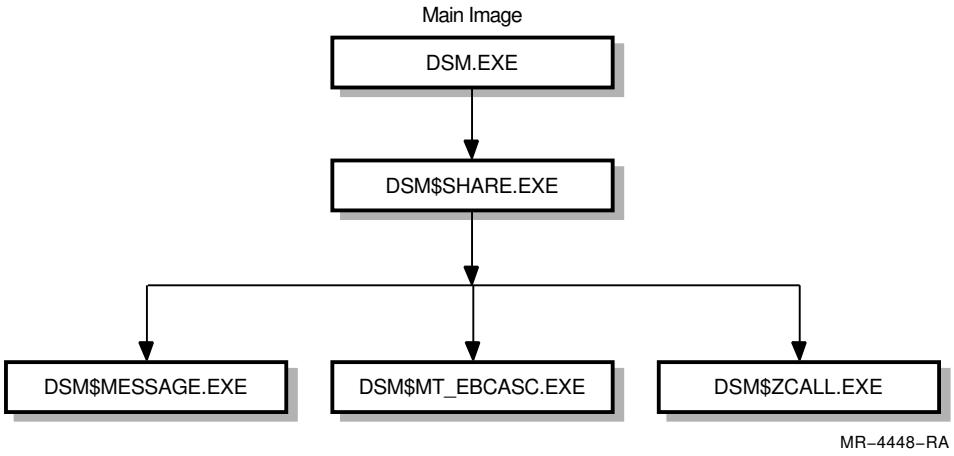
Example 1–1 (Cont.) VMSINSTAL Console Output

```
directory [SYSHLP.EXAMPLES]

 DSM$ADBOOK.PAS                -  Example of using the callable routines
 DSM$GLISTP.PAS                -  Example of using the callable routines
 DSM$ORDER.FOR                 -  Example of using the callable routines
 DSM$ZCALL_USER_TABLE.MAR      -  Template source file for USER ZCALL table
 DSM$ZCALL_EXAMPLE_TABLE.MAR   -  Example ZCALL Table Source file
 DSM$ZCALL_EXAMPLE_PKTAB.MAR   -  Example ZCALL Package Table Source file
 DSM$ZCALL_EXAMPLE_CODE.MAR    -  Example user-written ZCALL procedures
 DSM$ZCALL_EXAMPLE_BUILD.COM   -  Example user-written ZCALL build procedure

directory [SYSTEST]

 DSM$IVP.COM           - Residual Installation Verification Procedure


The following DASL files will be created in [SYSLIB]

 DSM$DASL_ROUTINES.SAV
 DSM$DASL_GLOBALS.SAV
 DSM$DASL_DEMO.SAV

The following DASL files will be created in [SYSHLP.EXAMPLES]

 DSM$DASL_PRTD_ROU.SAV
 DSM$DASL_PRTD_GLO.SAV

%VMSINSTAL-I-RESTORE, Restoring product saveset C ...
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...


        Calling VAX DSM Installation Verification Procedure


         VAX DSM V6.0
           SYSTEM
         [Baseline]


         VAX DSM V6.0 IVP Successful


%DSM-I-HALT, HALT command executed

VAX DSM installation is complete.  You may enter the following
command line to initialize a manager account for use with VAX DSM:

                  @SYS$MANAGER:DSM$INIT

        Installation of DSM V6.0 completed at 09:04

Enter the products to be processed from the next distribution volume set.
* Products: EXIT

        VMSINSTAL procedure done at 09:04
```

VAX DSM Shareable Images

VAX DSM is implemented as one *main image* and several VMS *shareable images*. VMS shareable images allow multiple programs to call the same image. This organization reduces memory utilization and increases system performance. The relationship of the various images included with VAX DSM is described in Figure 1–1.

Figure 1–1 VAX DSM Image Relationships



- DSM.EXE

  DSM.EXE is the main entry routine for all programmer and application users of VAX DSM. It accepts the DSM command line qualifier and then calls the DSM$SHARE image to process the qualifier and initiate access to VAX DSM.

- DSM$SHARE.EXE

  DSM$SHARE.EXE is the shareable, position-independent, image that supplies the entry points for the callable interface. A majority of the code that supports VAX DSM capabilities resides within this image. External programs that interface to VAX DSM link against this image. The DSM.EXE image is an example of an external program that calls VAX DSM through the callable interface located in DSM$SHARE.EXE.

- DSM$MESSAGE.EXE

  DSM$MESSAGE.EXE holds the VAX DSM error message text. VAX DSM Version 6.0 includes the message text as a separate image so that external callable interface programs can have access to VAX DSM error messages.

- DSM$MT_EBCASC.EXE

  DSM$MT_EBCASC.EXE holds the VAX DSM ASCII/EBCDIC conversion tables that are used by the VAX DSM magnetic tape handler for reading and writing IBM formatted magnetic tapes. You can create a customized ASCII/EBCDIC conversion table to meet your site requirements.

- DSM$ZCALL.EXE

  DSM$ZCALL.EXE is the ZCALL package table image. This image describes the existence of all user written $ZCALL packages.

Installing VAX DSM as Shared Images

The supplied VAX DSM images must be installed as VMS shared images in order to run VAX DSM. This is accomplished using the VMS Install Utility.

If you ran the IVP during kit installation, the images are installed as shared when the IVP executes. If you did not run the IVP, you must install all of the VAX DSM images as *shared images*. This reduces memory utilization, enhances VAX DSM image activation time, and applies required image privileges.

Use the SYS$STARTUP:DSM$INSTALL.COM procedure to install the VAX DSM images as shared images by logging in to a privileged VMS account that has the VMS CMKRNL privilege. Run the DSM$INSTALL command procedure by entering the following command:

```
$ @SYS$STARTUP:DSM$INSTALL Return
```

Installing VAX DSM in a VAXcluster Environment

If you are installing VAX DSM on a node in a VAXcluster environment, you must log in to the VMS system manager's account on each node that will be running VAX DSM and do the following:

1. Load the VAX DSM Product Authorization Key (PAK) by entering the following:

   ```
   $ LICENSE LOAD DSM Return
   ```

   For additional information on using LMF, see the *VMS License Management Utility Manual* or type HELP LICENSE at the DCL prompt.

2. Install VAX DSM as a shared image by entering the following:

   ```
   $ @SYS$STARTUP:DSM$INSTALL Return
   ```

Installation of the VAX DSM images using the DSM$INSTALL command procedure requires minimum SYSGEN parameter values for GBLPAGES and GBLSECTIONS. If you are unable to successfully run DSM$INSTALL on all VAXcluster nodes, you have to modify these SYSGEN parameters on the affected node and reboot that node. To do this, follow the procedure described earlier in this chapter in the section ''Determining SYSGEN Requirements for Installation.''

You can also use the VMS System Management Utility (SYSMAN) to perform these steps from your local node by defining your management environment to be a particular node, a group of nodes, or a cluster. See the *VMS SYSMAN Utility Manual* for more information.

Postinstallation Activities

Once VAX DSM installation is complete, you can create one or more VAX DSM environment manager accounts for use with VAX DSM *configurations*. The configuration completely controls access to global databases, DSM routines, and other VAX DSM capabilities.

This section provides detailed instructions for all activities required to create a VAX DSM environment and initialize a VMS account as a VAX DSM environment manager's account. Procedures for creating and starting a VAX DSM configuration are also briefly described. See Chapter 2 for complete information about VAX DSM configurations.

Creating the VAX DSM Environment Manager's Account

To define and manage a VAX DSM environment for development and applications, you must designate a VMS account for use as a VAX DSM *environment manager account*. The account provides a working area for enabling various VAX DSM options and operations. Using this account, the VAX DSM manager can perform tasks such as the following:

- Creating and extending VAX DSM database volume sets
- Creating and editing VAX DSM configurations
- Managing UCIs, globals, and routines on VAX DSM volume sets
- Starting up and shutting down VAX DSM configurations interactively

You use the SYS$MANAGER:DSM$INIT.COM command procedure to prepare a VMS account for use as a VAX DSM environment. To do this:

1. Log in to a privileged system account.
2. Run the DSM$INIT command procedure. Enter the following:

```
$ @SYS$MANAGER:DSM$INIT Return
```

The DSM$INIT command procedure allows you to create a new VMS account for use as a VAX DSM environment. You can also use DSM$INIT to designate an existing VMS account as a VAX DSM environment.

The DSM$INIT command procedure does the following:

- Asks you for the account name, UIC, password, login device, and log file directory of the VAX DSM manager account you want to create.
- Creates default directories to hold volume sets and journal files (before-image and after-image journal files) associated with this environment.
- Creates an account initialization DCL command procedure to do the following:
  - — Modify the user authorization file (UAF) to add the requested account
  - — Create the account login directory
  - — Create all directories required for use by VAX DSM configurations
  - — Create and grant a VMS *identifier* for the VAX DSM environment manager
  - — Create a VMS identifier for use by VAX DSM environment users
  - — Apply *access control lists* (ACLs) to all directories and files created

When the account initialization command procedure is completely created, DSM$INIT displays the procedure on your terminal. You can then optionally select to run the initialization command procedure. If you do not run the procedure, it is saved in a file with the following name format:

DSM$ADDMAN_ACCOUNT.COM

where:

ACCOUNT is the name of the manager account being created

If you choose to run the initialization command procedure, all of the operations that are displayed are performed. When the initialization procedure completes successfully, DSM$INIT takes the following actions:

- It creates an entry in the ^DSMMGR system global describing the manager account for the newly initialized environment.
- It creates the ^SYS environment configuration global in the directory you designated as the VAX DSM environment root directory. The ^SYS global is a VAX RMS™ ISAM-based global that is used by DSM utilities to define and start VAX DSM configurations.

- Creates the ^DSMACC configuration access global in the VAX DSM environment root directory. The ^DSMACC global is an RMS ISAM-based global used to store environment and DDP node access authorization information.

You can run the DSM$INIT procedure multiple times on the same account. DSM$INIT only makes the modifications necessary to produce a valid VAX DSM environment. In a VAXcluster environment, run DSM$INIT again whenever a new node is added to the cluster. This operation creates the required subdirectories for the new node. You can also rerun DSM$INIT to initialize new journal or database directories on additional disk drive units.

Running DSM$INIT.COM

Example 1–2 shows how to run the DSM$INIT.COM procedure to create a new VMS account and initialize it as a VAX DSM environment manager account. Note that this example is run in a VAXcluster environment.

Note: You must be logged in to a privileged system account in order to run the DSM$INIT command procedure.

Example 1–2 Running the DSM$INIT Command Procedure

```
$ @SYS$MANAGER:DSM$INIT Return

                    VAX DSM V6.0
            Environment Manager Initialization


  The VAX DSM environment manager must have a VMS account
  to define and control VAX DSM Configurations. This procedure
  allows you to create a new account or upgrade an existing
  VMS account as a VAX DSM environment manager.


Enter the VMS account name to use with VAX DSM  >DSMMGR Return

The DSMMGR account does not currently exist
Do you want to create and initialize this account [Y OR N] ?<Y> Return


Each VMS account must have a UIC identifier associated with it.
A UIC is in the form:

[group,member]

The group is an octal number less than 37777 and the
member is an octal number less than 177777. Enter the
UIC for the VAX DSM environment manager's account:
```

Example 1–2 Cont'd on next page

Example 1–2 (Cont.) Running the DSM$INIT Command Procedure

```
Enter the UIC for this account    >[250,500] Return


  An account password is required in order to add the account.
  Always select a password that is not obvious or easy to guess.

Enter the account password (not echoed)    > Return
Enter the same password again for verification    > Return

  Each VMS account requires a default login directory.
  This procedure creates an account with a login device
  and directory name. The login directory name is always
  equivalent to the account name.


  Enter the login device for the VAX DSM environment manager's
  account:

Enter the login device for this account    >DISK$USER Return

        VAX DSM requires a VMS directory to store log files,
        routine map files, RMS globals and RMS routine files to
        serve as the VAX DSM Environment Root Directory. This
        directory also holds subdirectories which are default
        storage locations for VAX DSM volume sets and journal files.

        You can select any directory or device as the VAX DSM
        environment root directory.

        By default, new VAX DSM environment directories are created
        as a tree of directories whose root is the manager's login
        directory. A VMS file protection scheme is applied to all of
        these directories in order to control access to database,
        journal, and other files used by the environment.


Enter the directory to use for VAX DSM <DISK$USER:[DSMMGR]> Return

        Use of VAX DSM in a VAXcluster environment requires
        the creation of a separate subdirectory for each cluster
        member. The subdirectories hold node-specific log files
        created by VAX DSM configurations running on each of the
        VAXcluster nodes. Appropriate subdirectories will be created
        as part of this initialization process. In order to do so,
        all cluster nodes must be identified at this time.


The following VAXcluster nodes are currently booted and will be
initialized for use with VAX DSM:

CASEY   ZORBA
```

Example 1–2 Cont'd on next page

Example 1–2 (Cont.) Running the DSM$INIT Command Procedure

```
Are there any additional Cluster members not currently
booted that you would like to add to this list [Y OR N] ?<N>Return

        This initialization process will create default directories
        to hold database and journal files associated with this
        environment. The default directories are created as
        subdirectories of the environment root directory.

        The following directories will be created if they do not
        already exist:


          DISK$USER:[DSMMGR.DATABASE]
          DISK$USER:[DSMMGR.JOURNAL]


        You may also want to create default directories for database
        and journal files on disk units other than the one used for
        the account login directory. You can optionally select to
        initialize these additional directories at this time.


Create similar directories on other disk devices [Y OR N] ?<N>Y Return

  Enter the disk device name that will receive the new
  directories. The disk unit must currently be available
  to successfully initialize the directories. Enter ^L
  to see a list of disk devices currently mounted on your system.


Enter an alternate disk for database/journal directories >DISK$DATA Return
Enter an alternate disk for database/journal directories > Return

The following command procedure has been created to update
the new VAX DSM manager's account and directories:
================================================================================
$ !         Command Procedure to Initialize DSMMGR
$ !         for use as a VAX DSM manager's account
$ !
$ IF P1 .NES. "" THEN GOTO CONTINUE
$ WRITE SYS$OUTPUT "You MUST enter with P1 = account password"
$ EXIT
$ !
$ CONTINUE:
$ !
$ ON ERROR THEN GOTO ERROR
$ UAF = "$AUTHORIZE"
$ IF F$TRNLNM("SYSUAF") .EQS. "" THEN $ DEFINE SYSUAF SYS$SYSTEM:SYSUAF.DAT
$ WRITE SYS$OUTPUT "Adding new Account for use as a VAX DSM environment manager . . ."
$ !
$ ! Add account and set password lifetime to 90 days, preexpired
$ !
$ UAF ADD DSMMGR/PASSWORD='P1/UIC=[250,500]/OWNER=VAXDSM -
                /DEVICE=DISK$USER:/DIR=[DSMMGR] -
                /FLAGS=NODISUSER -
                /PWDEXPIRED/PWDLIFETIME=90-00
```

Example 1–2 Cont'd on next page

Example 1–2 (Cont.) Running the DSM$INIT Command Procedure

```
$ !
$ WRITE SYS$OUTPUT "Modifying Environment Manager Account Quotas  . . ."
$ UAF MODIFY DSMMGR/ENQLM=1000
$ UAF MODIFY DSMMGR/BYTLM=40960
$ UAF MODIFY DSMMGR/ASTLM=36
$ UAF MODIFY DSMMGR/DIOLM=18
$ UAF MODIFY DSMMGR/WSDEF=500
$ UAF MODIFY DSMMGR/WSQUOTA=1000
$ UAF MODIFY DSMMGR/WSEXTENT=1200
$ UAF MODIFY DSMMGR/FILLM=150
$ UAF MODIFY DSMMGR/PRIV=TMPMBX
$ UAF MODIFY DSMMGR/DEFPRIV=TMPMBX
$ !
$ UAF SHO DSMMGR
$ !
$ READ/PROMPT="Account initialized - press return to continue" SYS$COMMAND A
$ !
$ UAF ADD/IDENTIFIER/ATTRIBUTE=RESOURCE DSM$MANAGER_DSMMGR
$ ADD/IDENTIFIER DSM$USER_DSMMGR
$ UAF GRANT/IDENTIFIER DSM$MANAGER_DSMMGR DSMMGR
$ !
$ !         Setup environment root directory
$ !
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$USER:[DSMMGR]
$ !
$ SET DIRECTORY                         DISK$USER:[DSMMGR] -
     /ACL=((DEFAULT_PROTECTION,S:RWED,O:RWED,G,W),-
           (IDENTIFIER=DSM$MANAGER_DSMMGR,OPTION=DEFAULT,-
            ACCESS=READ+WRITE+EXECUTE+DELETE),-
           (IDENTIFIER=DSM$MANAGER_DSMMGR,ACCESS=READ+WRITE+EXECUTE))
$ !
$ !
$ !         Setup cluster node directories
$ !
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$USER:[DSMMGR.CASEY]
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$USER:[DSMMGR.ZORBA]
$ !
$ !         Setup database/journal directories
$ !
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$DATA:[DSMMGR]
$ SET DIRECTORY                         DISK$DATA:[DSMMGR] -
     /ACL=((DEFAULT_PROTECTION,S:RWED,O:RWED,G,W),-
           (IDENTIFIER=DSM$MANAGER_DSMMGR,OPTION=DEFAULT,-
            ACCESS=READ+WRITE+EXECUTE+DELETE),-
           (IDENTIFIER=DSM$MANAGER_DSMMGR,ACCESS=READ+WRITE+EXECUTE))
$ !
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$DATA:[DSMMGR.DATABASE]
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$DATA:[DSMMGR.JOURNAL]
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$USER:[DSMMGR.DATABASE]
$ CREATE/DIRECTORY/LOG/OWN=[250,500]/PROT=(S:RWE,O:RWE,G,W) -
                                        DISK$USER:[DSMMGR.JOURNAL]
$ !
$ ERROR:
$ EXIT
================================================================================
```

Example 1–2 Cont'd on next page

Example 1–2 (Cont.) Running the DSM$INIT Command Procedure

```
Do you want to edit this command procedure [Y OR N]?<N> Return

  Type YES if you want to make modifications to this command
  procedure before it is run. The EDT™ editor is called to allow
  you to edit the file. Press Return if the command procedure is
  satisfactory.

  Note that you should use care in reducing account parameters
  and privileges, or in modifying how directories are created.
  The proper initialization of the account depends on these
  account parameters.

  You can save the command procedure and run it at a later time
  by answering NO to the following question:


Do you want to run this command procedure now [Y OR N] ?  >Y Return

Adding new Account for use as a VAX DSM environment manager . . .
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier DSMMGR value: [000503,000001] added to
rights data base
Modifying Environment Manager Account Quotas  . . .
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
```

The updated account authorization record is now displayed.

```
Username: DSMMGR                            Owner:  VAXDSM
Account:                                    UIC:    [250,500] ([DSMMGR])
CLI:      DCL                               Tables: DCLTABLES
Default:  DISK$USER:[DSMMGR]
LGICMD:   LOGIN
Login Flags:
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
No access restrictions
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:         90 00:00    Pwdchange:      (pre-expired)
Last Login:            (none) (interactive), (none) (non-interactive)
Maxjobs:         0  Fillm:       150  Bytlm:        40960
Maxacctjobs:     0  Shrfillm:      0  Pbytlm:           0
Maxdetach:       0  BIOlm:        18  JTquota:       1024
Prclm:           2  DIOlm:        18  WSdef:          500
Prio:            4  ASTlm:        36  WSquo:         1000
Queprio:         0  TQElm:        10  WSextent:      1200
CPU:        (none)  Enqlm:      1000  Pgflquo:      10240
Authorized Privileges:
  TMPMBX NETMBX
Default Privileges:
  TMPMBX NETMBX
```

Example 1–2 Cont'd on next page

Example 1–2 (Cont.) Running the DSM$INIT Command Procedure

```
Account initialized - press return to continue
%UAF-I-RDBADDMSG, identifier DSM$MANAGER_DSMMGR value: %X8001000F added
to rights data base
%UAF-I-RDBADDMSG, identifier DSM$USER_DSMMGR value: %X80010010 added to
rights data base
%UAF-I-GRANTMSG, identifier DSM$MANAGER_DSMMGR granted to DSMMGR
%CREATE-I-CREATED, DISK$USER:[DSMMGR] created
%CREATE-I-CREATED, DISK$USER:[DSMMGR.CASEY] created
%CREATE-I-CREATED, DISK$USER:[DSMMGR.ZORBA] created
%CREATE-I-CREATED, DISK$DATA:[DSMMGR] created
%CREATE-I-CREATED, DISK$DATA:[DSMMGR.DATABASE] created
%CREATE-I-CREATED, DISK$DATA:[DSMMGR.JOURNAL] created
%CREATE-I-CREATED, DISK$USER:[DSMMGR.DATABASE] created
%CREATE-I-CREATED, DISK$USER:[DSMMGR.JOURNAL] created

      The DSM management globals will now be updated . . .

Updating DSM Management Global:         ^[SYS$LIBRARY]DSMMGR
Initializing DSM Configuration Global:  ^[DISK$USER:[DSM$DSMMGR]]SYS

Account DSMMGR successfully initialized

The account has been created with your specified password.
However, for security reasons, the password has been set to expire.
Please log in to the new account immediately and reset the account
password.

You may now log in to the VAX DSM manager's account and
use the SYS$MANAGER:DSM$CONFIGURE.COM command procedure to
initialize VAX DSM database volume sets, or create and start up
new VAX DSM configurations.

See the "VAX DSM Installation and Management Guide" for complete
information about VAX DSM configurations and the "VAX DSM Database
Operations Guide" for complete information about database volume
sets.
```

Creation of the VMS account and its initialization as a VAX DSM manager account is now complete. If you are running within a VAXcluster environment, DSM$INIT initializes all nodes for use with VAX DSM in a single operation.

The process quotas and privileges used to create the VAX DSM manager account in the previous example are generally good values for initial use. You may find that you want to adjust these values as your requirements change. It is important to set the values for WSQUOTA and WSEXTENT appropriately to ensure good performance. However, recommendations for these values depend entirely on the characteristics of the configuration that you intend to run. After you complete installation and initialization, read Chapter 2 and Chapter 4 carefully. They provide details on how you can establish the appropriate values for the VAX DSM manager's and user's accounts, as well as how to tailor VMS SYSGEN parameters for your site.

For details on the VMS Authorize Utility, see the *VMS Authorize Utility Manual*.

The DSM$INIT command procedure grants only the VMS TMPMBX privilege during account creation and applies the VMS default UAF record values. However, your site may require the OPER privilege. OPER is required to use the ZUSE MUMPS command, or operator REPLY/TO command for coordinating after-image journal file usage.

Configuring VAX DSM

Once the VAX DSM manager account is initialized, you must log in to the manager account and run the SYS$MANAGER:DSM$CONFIGURE.COM procedure. The DSM$CONFIGURE command procedure provides a simple introduction to the methods used to create VAX DSM databases and define VAX DSM configurations. Once you become more familiar with these techniques, you may choose to directly call standard VAX DSM utilities to manage your configurations. The DSM$CONFIGURE command procedure optionally allows you to:

- Create a new VAX DSM database volume set.
- Define a VAX DSM configuration.
- Start a VAX DSM configuration in the newly initialized VMS account.

To use DSM$CONFIGURE, log in to the VAX DSM environment manager account and enter the following:

```
$ @SYS$MANAGER:DSM$CONFIGURE Return

            VAX DSM Configuration Management Procedure

This procedure is intended to be used while logged in to a newly
initialized VAX DSM manager account to configure and start DSM
for the first time. It serves as an introduction to the techniques
used to build VAX DSM database volume sets, create new DSM
configuration definitions, and start a fully functional VAX DSM
configuration. The following VAX DSM utilities are called:

        ^VOLMAN - creates new VAX DSM database volume sets
        ^CONFIG - builds  VAX DSM configuration definitions
        ^STU    - starts a VAX DSM configuration

You can independently access any of these utilities at a later time
to further modify the VAX DSM environment. See the VAX DSM
Installation and Management Guide for complete information about
DSM configurations. See the VAX DSM Database Operations Guide for
information about database sets and volume sets.


Do you want to create a new VAX DSM database volume set [Y OR N] ?<Y> Return

You can elect to build a VAX DSM database volume set at this time.
If you want to create a new volume set answer YES. The VAX DSM
Database Operations Guide contains complete information on VAX DSM
volume sets and the procedure used here to create new volume sets.
```

```
Initialize a VAX DSM database Volume Set

Every DSM volume set must have a unique 3-character identifier.
Enter the 3-character name for the volume set >EMP Return

Create a DSM Volume in which file ?
    <DISK$USER:[DSMMGR.DATABASE]EMPVOL1.GLS> Return

Enter desired size of this volume in 1 KB VAX DSM blocks >20000 Return

Maximum number of globals allowable on this volume set <90> Return
Allow expanded string lengths on this volume set [Y OR N] ? <N> Return

What would you like the new label on this volume to be ?
(up to 22 characters, enclosed in quotes)   >"EMP Volume 1" Return

Initializing DISK$USER:[DSMMGR.DATABASE]EMPVOL1.GLS
for use as a VAX DSM volume...

DISK$USER:[DSMMGR.DATABASE]EMPVOL1.GLS initialized as
Volume 1 of VAX DSM volume set EMP with 20000 1 KB blocks (50 DSM maps)

               Build VAX DSM configuration

               Type ? for HELP at any time

Would you like extended help [Y OR N] ? <N> Return

Enter the configuration identifier <1>PAYROLL Return

Enter the name of the configuration that you want to create.
See Chapter 2 for complete information on VAX DSM configurations
and the procedure used here to create a configuration.
.
.
.
Start up configuration PAYROLL [Y OR N] ? <Y> Return
.
.
.
```

Once you create the startup configuration, you can start VAX DSM. The startup procedure establishes all processes necessary for your VAX DSM system environment. See Chapter 2 for information about these processes.

If VAX DSM startup fails, an explanatory message is displayed on your terminal. Initialization failure is often due to improper VMS SYSGEN parameters such as global page quota, or improper VAX DSM manager account quotas. In many cases, the reason for failure is stored in the DSMDEMON.LOG file. To correct a failure, tailor SYSGEN parameters or account quotas as described in Chapter 4. The appropriate values for these parameters differ depending on the characteristics of the configuration you are attempting to start. The ^MEMREQ Utility provides these values for each configuration you build.

After you change these parameters, run AUTOGEN and reboot your VMS system before attempting to restart your VAX DSM configuration.

> **Note:** If you intend to use VAX DSM on multiple nodes of a cluster, you must define a node-specific configuration for each VAXcluster node and start the configuration on the target node. For more information on how to run VAX DSM in a VAXcluster environment, see the section ''Initializing VAX DSM in a VAXcluster Environment'' in Chapter 2 and see the *VAX DSM Database Operations Guide*.

Editing the VMS System Files

You need to edit the VMS system startup and shutdown files to provide for automatic startup and shutdown of VAX DSM configurations when your system is rebooted. The following sections describe the procedures for system startup and shutdown.

System Startup

The SYS$STARTUP:DSM$INSTALL.COM procedure is supplied as a template for use at VMS boot time to initialize VAX DSM. This procedure has examples of a number of activities that many sites need to do to enable VAX DSM capabilities. These activities include:

- Installing DSM images as shared
- Allocating physical pages for disk buffer usage
- Loading the VAX DSM DDP driver
- Defining search lists for DSM logical name tables
- Mounting DSM-11 compatible disks
- Creating a batch queue for DSM startup
- Starting up predefined VAX DSM configurations

The only activities initially enabled with the procedure provided is the installation of DSM images as shared and the definition of logical names. Examples of how to accomplish the remaining activities are provided, but the actual DCL commands used are disabled by preceding each command line with a comment character (!). The procedure has a detailed description of how to customize DSM$INSTALL for your site and enable the relevant capabilities. Once you have edited DSM$INSTALL, edit SYS$STARTUP:SYSTARTUP_V5.COM, the VMS system startup command procedure, to include the following command:

```
$ @SYS$STARTUP:DSM$INSTALL
```

System Shutdown

The SYS$MANAGER:DSM$SHUTDOWN.COM procedure shuts down a VAX DSM configuration. You must edit the system shutdown command procedure, SYS$MANAGER:SYSHUTDWN.COM, to call the DSM$SHUTDOWN command procedure and automatically shut down running VAX DSM configurations when your VMS system is shut down. Include the following command line in the SYS$MANAGER:SYSHUTDWN.COM file:

```
$ @SYS$MANAGER:DSM$SHUTDOWN p1
```

where:

p1 is the name of the VAX DSM environment manager's account

Include the previous command line in the SYS$MANAGER:SYSHUTDWN.COM procedure for each VAX DSM environment that is initialized. This ensures that any active VAX DSM volumes are dismounted in an orderly fashion before VMS shuts down.

> **Note:** If you fail to include @SYS$MANAGER:DSM$SHUTDOWN.COM in your system shutdown command file for each VAX DSM environment, database degradation may result.

Chapter 2
Managing VAX DSM

This chapter provides VAX DSM managers with information about managing a VAX DSM environment. It is assumed that you have already initialized a VMS account for use as a VAX DSM environment by following the procedures in Chapter 1. This chapter presents an overview of a VAX DSM environment and detailed information about VAX DSM configurations.

Chapter 5 contains information about optimizing the performance of VAX DSM. Experienced DSM programmers can find information to improve system performance in the *VAX DSM Programmer's Guide*.

Chapter 4 describes how to tune VMS for use with VAX DSM. For detailed information about volume sets and database integrity, see the *VAX DSM Database Operations Guide*. For additional information about VMS system management, see the *Introduction to VMS System Management* and the *Guide to Setting Up a VMS System*.

VAX DSM Environment

A VAX DSM *environment* is the working area used by a VAX DSM manager for the purposes of enabling and controlling the operations of VAX DSM. The initialization of an environment configures the following resources to support VAX DSM management:

- VMS account

  The VAX DSM environment manager's account provides a secure operational account from which the manager can control DSM. Account quotas are propagated to the various supporting VAX DSM system processes when VAX DSM is fully configured.

- VAX DSM directories

  VAX DSM directories and subdirectories are created for use by VAX DSM processes and are the default location for the storage of database and journal files.

- VAX DSM environment identifier

  A VMS *identifier* is created for the environment and granted to the VAX DSM manager's account. The identifier represents the privileges necessary to accomplish VAX DSM management functions within the environment. Holders of the identifier can create volume sets, define configurations, and control other operations within the environment.

- Environment configuration global (^SYS)

  The *environment configuration global* is an RMS based global named ^SYS (file name SYS.GBL). The ^SYS global holds the complete database of VAX DSM *configurations* that the VAX DSM manager has defined for the environment. A configuration describes the various enabled VAX DSM capabilities, available databases, and other parameters relevant to a fully operational VAX DSM environment. After initialization, ^SYS is generally empty.

- Configuration access global (^DSMACC)

  The *configuration access global* is an RMS global named ^DSMACC (file name DSMACC.GBL), The ^DSMACC global holds the list of VMS accounts that are authorized to access VAX DSM configurations running in this environment. After initialization, ^DSMACC is generally empty.

Environment Directories, Files, and Protection

When you initialize a VAX DSM environment using DSM$INIT.COM, a set of VAX DSM directories is created as a tree of directories. The directories are used to hold VAX DSM log files, mapped routines, and database files. The top-level directory of the tree is known as the *environment root directory*. By default, this directory is created as the environment manager's account login directory. However, DSM$INIT allows you to configure the environment root directory as a directory or subdirectory anywhere on your system.

Several subdirectories of the environment root directory are also created to support the environment:

- Default journal directory ([.JOURNAL])

  The journal directory is the default location for after-image or before-image journal files.

- Default database directory ([.DATABASE])

  The database directory is the default location for VAX DSM database volume sets.

- Cluster node directories

  In a VAXcluster environment, a separate subdirectory is also created for each member of the cluster. VAXcluster environments are discussed later in this chapter in the section ''Initializing VAX DSM in a VAXcluster Environment.''

The environment root directory holds the ^SYS environment configuration global and ^DSMACC configuration access global. The root directory is also the default location for mapped routine files created using the ^RMBLD Utility. In a single-node configuration, log files, error files, and option files for the DSM detached processes are also located within the root directory.

Database and journal subdirectories are default locations for the storage of VAX DSM volume sets and journal files. VAX DSM utilities that prompt for the location of a database or journal file always use one of these directories when creating a default response. You can always enter a different file specification to override the default. DSM$INIT optionally initializes additional database and journal directories on other system devices. For example:

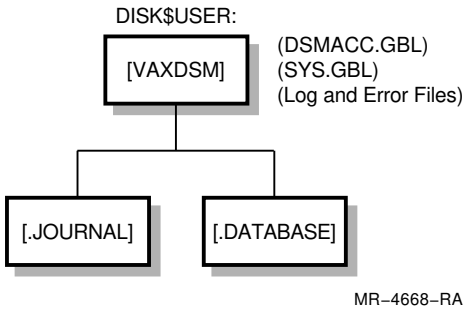
```
$ @SYS$MANAGER:DSM$INIT Return
  .
  .
  .
  This initialization process will create default directories
  to hold database and journal files associated with this
  environment. The default directories are created as
  subdirectories of the environment root directory.

  The following directories will be created if they do not
  already exist:

  DISK$USER:[DSMMGR.DATABASE]
  DISK$USER:[DSMMGR.JOURNAL]

  You may also want to create default directories for database
  and journal files on disk units other than the one used for
  the account login directory. You can optionally select to
  initialize these additional directories at this time.

Create similar directories on other disk devices [Y OR N] ?<N>Y Return

  Enter the disk device name that will receive the new
  directories. The disk unit must currently be available
  to successfully initialize the directories. Enter ^L
  to see a list of disk devices currently mounted on your system.
```

```
Enter an alternate disk for database/journal directories >DISK$DATA Return
Enter an alternate disk for database/journal directories > Return
.
.
.
```

Caution: Never place your volume set file on the same physical device as its after-image journal files. If the device fails, you need the after-image journal files to recover the backup copy of the volume set.

Figure 2–1 describes the relationship between the various directories in a single-node system.

Figure 2–1 Directory Hierarchy in a Single-Node System



VMS UIC-based protection is applied to all directories and RMS global files initialized by DSM$INIT. DSM$INIT also creates a unique VMS identifier for the new environment. The identifier is granted to the VAX DSM environment manager's account and is then associated with each of the initialized files through the application of a VMS access control list (ACL). Only the VAX DSM environment manager or the *alternate manager account* that holds the identifier can access files protected by this method.

Default type ACLs are also applied to the directories so that new files created within the directories receive the same UIC and identifier-based protection. The applied protections prevent unauthorized access to sensitive environment, configuration, database, and journal files. See Chapter 3 for complete details on how to control access to DSM environments and configurations.

The DSM$INIT procedure also creates a VMS identifier for use in protecting access to DSM-11 disk drives and the VAX DSM environment logical name table.

VAX DSM Baseline Mode

Once a VAX DSM environment has been initialized for use, you can log in to the environment manager's account and use VAX DSM in *baseline mode*. Use the DSM command with the /MANAGER command line qualifier to enter baseline mode:

```
$ DSM/MANAGER Return
```

This gives you direct access to the DSM command line interpreter. VAX DSM displays the following banner:

```
VAX DSM V6.0
   DSMMGR
 [Baseline]

>
```

The [Baseline] *access mode specifier* indicates that there is no configuration running. In baseline mode, VAX DSM offers only minimal services, and its primary function is intended for use in running VAX DSM utilities. Baseline access has the following characteristics:

- All VAX DSM interpreter capabilities are available.
- Globals are stored and accessible only in RMS globals.
- Routines are stored and accessible only in RMS routine directories.
- Journaling, Distributed Data Processing (DDP), and other advanced capabilities are not available.

In baseline mode, global and routine access is limited to utility globals and routine directories that are supplied by the VAX DSM installation procedure. VAX DSM managers use baseline mode for management activities such as:

- Creating VAX DSM volume sets
- Defining VAX DSM configurations
- Starting defined VAX DSM configurations

VAX DSM Configurations

A VAX DSM *configuration* is a database that describes the operational capabilities and characteristics of a fully functional VAX DSM environment. A configuration definition specifies the following:

- How many disk buffers are available for global data caching
- Which DSM database sets are included, and if the following are true:
  - — Cluster-mounting is enabled.
  - — Before-Image Journaling is enabled.
  - — After-Image Journaling is enabled.

- Which global references are translated
- Which mapped routines are installed
- If DDP is enabled
- The default values of the DSM command line qualifiers

As the VAX DSM environment manager, use the Create Configuration Utility (^CONFIG) while logged in to the baseline system to name and define the capabilities of a configuration. Once a configuration definition is complete, the configuration is *started* using the Configuration Startup Utility (^STU). Configurations can only be started while logged in to the VAX DSM environment manager account.

A configuration is considered *running* when all the defined configuration capabilities are available to VAX DSM environment users.

A running VAX DSM configuration includes a set of shared data structures and detached *VAX DSM system processes* that provide highly efficient access to global databases. The configuration establishes all capabilities available to DSM users.

When a VAX DSM system is running, programmers and application users can log in to it from various VMS accounts. Users can accept the defaults established by the configuration, or can override some of the defaults using qualifiers on the DSM command.

A configuration is considered *shut down* when only the baseline capabilities are available.

If a programmer or application user tries to run VAX DSM without the /MANAGER command line qualifier when no configuration is running, the user cannot access DSM. The system displays the following error message:

```
%DSM-E-NOTRUN A VAX DSM configuration is not running
```

VAX DSM Accounts

You can establish any number of VMS user accounts to provide access to a running configuration. There are three types of accounts that can be initialized for use with VAX DSM:

- VAX DSM environment manager account

  The DSM$INIT.COM procedure creates the VAX DSM environment managers account. VAX DSM can only be started up from this account. You can create or extend volume sets, or perform the other management tasks described in this chapter from this account. You can shut down VAX DSM from this account as well as the VMS system account.

  Alternate environment manager accounts for an environment can be designated by the environment manager using the Access Control Utility (^ACL).

- VAX DSM programmer accounts

  These accounts are for the interactive users who develop VAX DSM routines. Programmer accounts are created using the VMS Authorize Utility. Programmer access to DSM configurations can be controlled to limit access to specific VAX DSM volume sets and UCIs and disable access to VAX DSM management functions.

- VAX DSM application user accounts

  These accounts are for users executing only particular VAX DSM routines under controlled conditions. Application accounts are created using the VMS Authorize Utility. These accounts are configured so that application users are tied to the DSM application and cannot interact directly with the VAX DSM interpreter.

The section ''Establishing VMS Accounts for Programmers and Application Users'' in Chapter 3 describes how to set up the programmer and application user accounts.

Overview of VAX DSM Databases

VAX DSM databases are organized into named *database sets*. A database set is described by its three components:

- DSM volume sets

  DSM volume sets are specially formatted VMS files that hold DSM globals and routines.

- After-image journal files

  After-image journal files are used to keep an audit trail of modifications made to globals within a running configuration.

- Before-image journal file

  The before-image journal file maintains a record of the contents of a database block before a major database event occurs. Major database events include block splits caused by a global set operation or block merges caused by a global kill operation.

Before defining VAX DSM configurations, you initialize VAX DSM volume sets using the ^VOLMAN Utility. This operation preallocates VMS files and places them on system disk units in the various VAX DSM directories you created during environment initialization. Volume sets can be configured to consist of one to eight VMS files.

Once the volume sets you want to use in your application are created and placed, you can organize them into database sets. The database set definition includes all volume set file names and journal file names (after-image and before-image).

The following characteristics of the database set are also included in the definition:

- Cluster-mounted

  When this characteristic is enabled, the database set can be mounted and shared by multiple members in a VAXcluster environment.

- Transaction processing

  When this characteristic is enabled, related modifications to the database set can be organized into well-formed *transactions*. The transactions are defined within application program *recovery units*.

Three types of database sets are supported:

- *DSM-11 compatible database sets*

  The DSM-11 compatible database set includes one DSM-11 volume set mounted as VMS foreign volumes, optional after-image journal files, and an optional before-image journal file.

- *VAX DSM database sets*

  A VAX DSM database set consists of one VAX DSM volume set, optional after-image journal files, and an optional before-image journal file.

- *Transaction database sets*

  A transaction database set includes one or more volume sets, required after-image journal files, and a required before-image journal file. VAX DSM Version 6.0 transaction recovery unit processing only applies to globals resident within transaction database sets.

Each database set within a VAX DSM environment is uniquely named and defined. You use the Create/Modify Database Set Utility (MODIFY^CONDBS) to create and modify the components of a named database set.

When you create a configuration, you specify which database sets to include for mounting. To make database set definition easier, the ^CONFIG Utility also allows you to create and modify database sets as you define a configuration.

Once a database set is created, it can be included in multiple configurations. In a VAXcluster environment, you only need to define the database set once. Each cluster member can then name the same database set for mounting on a clusterwide basis within the configurations defined for that node. If you modify the components of a database set, the modifications are reflected in each configuration that references the database set.

For details about how to create and manage volume sets and database sets, see the *VAX DSM Database Operations Guide*.

Configuration Management (^CONMAN)

The ^CONMAN Utility provides utilities to create, copy, delete, display, and modify configurations and configuration information. Enter the following to run ^CONMAN:

```
>D ^CONMAN Return
```

The utility displays the following menu:

```
Configuration Management

    1.  COPY CONFIGURATION                  (COPY^CONFIG)
    2.  CREATE CONFIGURATION                (^CONFIG)
    3.  DELETE CONFIGURATION                (DELETE^CONFIG)
    4.  DISPLAY CONFIGURATION               (DISPLAY^CONFIG)
    5.  MODIFY EXISTING CONFIGURATION       (MODIFY^CONFIG)

Select Option >
```

The following section describes the ^CONFIG Utility. For more information on the other ^CONMAN utilities, see Chapter 6.

Defining Configurations (^CONFIG)

Use the Create Configuration Utility (^CONFIG) to define new configurations or modify the characteristics of an existing configuration. The ^CONFIG Utility stores configurations by a *configuration identifier* that you designate. You can create and store any number of configurations. Once a configuration is stored, you can retrieve it, edit its characteristics, and then start the configuration.

To create a new VAX DSM configuration, log in to the VAX DSM environment manager's account and run VAX DSM with the /MANAGER command line qualifier. Example 2–1 explains the various configuration options available and shows how to create a new configuration. References to more detailed documentation are provided for each question.

> **Note:** Pressing Ctrl/c while using ^CONFIG aborts the utility. This leaves the configuration partially defined and inconsistent, and subsequent attempts to start the configuration may fail. Always answer all of the questions presented by the utility to completely define a configuration.

Example 2–1 Running the ^CONFIG Utility

```
$ DSM/MANAGER Return

          VAX DSM V6.0
             DSMMGR
           [Baseline]


>D ^CONFIG Return

Build VAX DSM Configuration

      Type ? for HELP at any time


Would you like extended help [Y OR N] ?<N>Y Return

Enter YES if you want help text displayed for every question asked.
Press Return to continue without extended help.


Enter the configuration identifier <1>PAYROLL Return

Enter the name of the configuration that you want to create,
or modify. You will use this name whenever you want to refer
to this configuration.

The name must consist of less than 12 uppercase alphabetic and
numeric characters and cannot be the number 0.

Type ^L to see a list of currently defined configurations.


PAYROLL is not currently defined, define it now [Y OR N] ? <Y> Return
Initializing PAYROLL as a new configuration...


Enter the number of disk buffers   >1200 Return

Disk buffers are 1KB in-memory buffers used to cache database
disk data in memory. The buffer techniques used by VAX DSM can
significantly reduce physical disk I/O and increase system
performance if the proper number of buffers is chosen. In
environments with a large number of users, it is not uncommon
to configure anywhere between 2000 to 6000 buffers. However,
memory resources can limit this value.

If you are creating a configuration for limited use and testing,
choose a small number (300-600). This reduces the amount of process
and VMS system tuning required to get the configuration started.
If you are creating a configuration for production use, it is
advisable to increase this value significantly and enable the use
of nonpaged memory.

A minimum of 64 buffers is allowed.

Reference: "Configuring VAX DSM Shared Memory" in Chapter 4
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Use nonpaged memory for these buffers [Y OR N] ?  <N>Return

VAX DSM can use nonpaged memory to contain the disk buffers that
are used for shared global access. Because disk buffers are
shared between all VAX DSM users and require a significant amount
of memory. Using nonpaged memory greatly reduces the amount of page
fault activity each VAX DSM process incurs and also reduces process
working set sizes. If you are creating a stable production environment
with greater than 600 disk buffers, it is advisable to use nonpaged
memory.

In order to use nonpaged memory, you must preallocate enough system
physical memory to contain the disk buffers and other shared data
structures for this configuration using the /PHYSICAL_PAGES=nnnn
command line qualifier. Use the ^MEMREQ Utility to determine the
nonpaged physical memory requirements for this configuration.

Enter YES if you have allocated nonpaged memory and want to use it
for the disk buffer cache. Answer NO if you want to use paged VMS
shared global sections for buffers and shared data structures.

Reference: "Configuring VAX DSM Shared Memory" in Chapter 4


Enter the maximum number of DSM users  <128>Return

Enter the maximum number of DSM users that you want to allow
simultaneous access to this configuration. Note that the Garbage
Collector, Write Demon, DDP processes, Recovery Process, and
Journal Process are all considered DSM users.

It is important to keep this number as small as possible because
additional memory usage is required to support each potential user.
Three is the minimum number of users allowed.


Enable configuration access control   [Y OR N] ? >YReturn

Configuration access control provides a complete security system for a
running configuration. VAX DSM defines three modes of access to VAX DSM:

      Manager Mode     - use of the /MANAGER qualifier
      Programmer Mode  - access as a programmer
      Application Mode - access as an application user

If you enable access control, you can use the ^ACL Utility to specify
which VMS users are permitted to log in to VAX DSM for each of these
access modes. You can restrict programmers to specific UCIs and restrict
application users to specific routines. In addition, Programmer Mode users
are not permitted to use the VIEW command, $VIEW function, privileged
system $ZCALL functions, or the /BYPASS command line qualifier.

If you disable access control, any VMS user can access any UCI in
Programmer Mode or Application Mode. Programmer Mode users have
unrestricted access to the VIEW command, $VIEW function, privileged
$ZCALL functions, and the /BYPASS command line qualifier.

Reference: "Configuration Access Control" in Chapter 3
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Include support for Routine/Global Performance Histograms [Y OR N]?<N>Return

The ^RTH Utility produces detailed reports and histograms on
the performance of your application within the VAX DSM
environment. Frequency of access and modification to specific
globals is reported along with statistics on routine calls and
CPU utilization. You must reserve additional memory at DSM
startup for purposes of ^RTHIST data collection.

Enter YES if you intend to use the ^RTH Utility. Enter NO if you
do not want to use the ^RTH Utility.

Reference: "Performance Statistics (^RTH)" in Chapter 5

***System options updated


Number of database sets in this configuration <1>Return

Enter the number of database sets for this configuration to mount
during startup. A configuration must mount at least one database set
that includes at least one volume set.  However, the total number
of volume sets defined by all of the database sets included within
the configuration cannot exceed 16.


Name of database set #1   >EMPLOYEE Return

Enter the name of a database set for this configuration to mount
during startup.

Enter ^L to get a list of the currently defined database sets within
the environment.


EMPLOYEE not defined. Define it now [Y OR N] ? <Y>Return

A configuration can only reference database sets that are already
defined. This database has not been previously defined.

Enter YES to define the database set now. Enter NO to go back and
enter a different database set name for this configuration.


Is this a VAX DSM, Transaction or DSM-11 database set [V, T OR D]<V>Return

You can designate this database set as one of three types of
database sets:

  DSM-11        volume set is comprised of DSM-11 foreign mounted
                disk volumes
  VAX DSM       volume set is comprised of VAX DSM volume files
  Transaction   comprised of one or more associated VAX DSM volume
                sets that support transaction processing updates
                to resident globals

Press Return to create a VAX DSM database set. Enter T or D to
create a Transaction or DSM-11 database set.
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Mount EMPLOYEE for clusterwide access  [Y OR N] ? <N>Return

If all of the volume sets in the database set are to be mounted by
two or more concurrently running configurations within a VAXcluster
environment, it must be declared as a cluster-mounted database set.
Cluster-mounted database sets allow READ/WRITE access for all of the
volume sets within this database set across a VAXcluster environment.

If this database set is to be mounted by only a single running
configuration, it should not be declared as cluster mounted.

Enter YES to cluster-mount this database set. Enter NO if you do
not want to cluster-mount this database set.

Reference: See the VAX DSM Database Operations Guide


Name of the volume set   >EMP Return

Enter the name of the volume set within this database set that is
to be mounted at configuration startup. The DSM volume set name is
the name given to the volume set when it was prepared using the
^VOLMAN Utility.

DSM volume set names must be 3-uppercase alphabetic characters.

Reference: See the VAX DSM Database Operations Guide


Enter the number of volumes in volume set EMP <1>Return

VAX DSM volume sets can consist of one to eight disk or file
volumes. Enter the number of volumes in this volume set.

Reference: See the VAX DSM Database Operations Guide


Enter the file name of EMP volume #1
<DISK$USER:[DSMMGR.DATABASE]EMPVOL1.GLS>Return

Enter the VMS file specification of the volume to mount during
configuration startup.

To override the default file specification supplied in brackets,
you only need to enter that portion of the default that you want
to modify. You can selectively override the disk, directory, filename,
or file extension. For example, to override the disk unit, enter a
new disk unit.

Reference: See the VAX DSM Database Operations Guide
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Enable Before-Image Journaling for this database set [Y or N] ? <N>Return

Before-Image Journaling saves separate copies of physical database
blocks as they appeared before a major modification of the database
is attempted. A major modification includes SET operations that
cause a new block to be allocated (split) or KILL operations that
cause the deallocation of a database block (merge).

This approach to journaling provides a high performance safeguard
against database degradation if your system should fail with
unwritten, modified blocks remaining in the disk buffer cache.
Because only a few blocks are journaled, and the before-image
journal file is continually recycled, restoration of a
database is extremely fast.

Enter YES to enable physical Before-Image Journaling for the
volume set in this database set. Enter NO if you do not want to
use Before-Image Journaling.

Reference: See the VAX DSM Database Operations Guide


Enable After-Image Journaling for this database set [Y or N] ? <N>Return

After-Image Journaling is a facility that allows you to save
critical global database transactions to a sequential disk file.
After-Image Journaling saves a separate physical record of all
SET and KILL operations that occur to the database. This trace
can be used in conjunction with BACKUP and RESTORE to completely
recover a database to its current state should a media failure
(for example, disk failure) occur.

Answer YES to include After-Image Journaling support for all
of the volume sets in this database set. Enter NO if you do not
want to include support for After-Image Journaling.

Reference: See the VAX DSM Database Operations Guide
*** Database set EMPLOYEE updated


Include support for Distributed Data Processing [Y OR N] ?<N>Return

Distributed Data Processing (DDP) is a facility that allows you to
access global arrays on another VAX DSM or DSM-11 system connected
using DEUNA/DEQNA Ethernet controllers. A routine can request data
from a remote system using an extended global reference syntax, or
global translation.

Enter YES to include support for DDP. Enter NO if you do not want
to include support for DDP.

Reference: See the VAX DSM Database Operations Guide

*** DDP parameters updated
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Modify global characteristics [Y OR N] ? <N>Return

Default global characteristics are applied to all newly created
globals on a configuration-wide basis. While these are the default
characteristics, you can still create globals with different
characteristics using the ^%GLOMAN Utility.

Enter YES if you want to modify this configuration's default global
characteristics. Enter NO if you do not want to modify the default
global characteristics.

Reference: See the VAX DSM Database Operations Guide and
           "Global Access Control" in Chapter 3


Include support for mapped routines [Y OR N] ?  <N>Return

Mapped routines is a facility that allows you to lock a set of DSM
application routines into virtual memory. These routines are mapped
and referenced by applications' users, thereby reducing the system
overhead that is usually required to load a disk resident copy of
a routine into a user's process space.

Mapped routine sets must first be created with the ^RMBLD Utility.
Once created, you can use this utility to name those routine sets
you want to have installed in memory during the startup procedure.

Enter YES to include mapped routine sets. Enter NO if you do not
want to include mapped routine sets.

Reference: "Using Mapped Routines" in Chapter 5

*** Mapped routine information updated


Modify DSM default command line qualifiers [Y OR N] ? <N>Return

The DSM command accepts command line qualifiers that are used to
define a specific DSM environment. The following are the command line
qualifiers that you can modify:

   /BYPASS                          /SOURCE_BUFFER_SIZE=n
   /[NO]CENABLE                     /STACK_SIZE=n
   /EXTENSION=(keyword[,...])       /SYMBOL_TABLE_SIZE=n
   /[NO]MAPPED                      /TERMINAL_BUFFER_SIZE=n
   /[NO]PACK                        /TYPEAHEAD
   /[NO]PASSALL                     /UCI=name
   /ROUTINES=[UCI,VOL]              /VOLUME_SET_NAME=name

Enter YES if you want to modify the values of the default command
line qualifiers for this configuration. Enter NO if you do not
want to modify these values.

Reference: See the VAX DSM Programmer's Guide
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Modify Global Translation Tables   [Y OR N]?<N>Return

The Global Translation Tables provide a flexible way to logically
force global references to occur in a non-default DSM directory (UCI,VOL).
It provides transparent access to globals between DSM directories on
any VAX DSM or DSM-11 system, local or remote (using DDP).

Reference: See the VAX DSM Database Operations Guide

*** Configuration PAYROLL defined


Make PAYROLL the new default configuration [Y OR N] ? <N>YReturn

If you want to make this configuration the new default configuration
to use when starting up VAX DSM, enter YES. If you do not want to
make this the default configuration, enter NO.


Show configuration memory requirements [Y OR N] ? <N>Return

Enter YES to produce a comprehensive report on memory usage
and suggested VMS tuning parameters. The reports are generated
by the ^MEMREQ Utility. The ^MEMREQ Utility uses the configuration
data supplied to determine which VMS SYSGEN parameters must be
modified. It then displays the appropriate modifications. It also
suggests process working set sizes and quotas for users of this
configuration.

Reference: "Using the ^MEMORY Utility" in Chapter 4


Start up configuration PAYROLL [Y OR N] ?  <Y>Return

Enter YES if you want to start VAX DSM with all of the options you
have specified for this configuration. If you need to modify VMS
SYSGEN parameters, it is best to answer NO here, make the necessary
modifications and reboot your system. You can then log back in to
VAX DSM and run the ^STU Utility to start the configuration.

The configuration is now completely defined and ready to be started.
If you answered YES to the previous question, the following question
appears:


Enable user logins on startup [Y OR N] ? <Y>Return

It is useful to disable logins in situations where system management
tasks require you to have exclusive access to a fully functional
VAX DSM configuration.

Answer NO to this question to disable user logins into VAX DSM after
startup completes. Only users logging in to VAX DSM with the /MANAGER
command line qualifier are allowed access to the running configuration.
All other users receive the "VAX DSM logins are currently disabled"
message.
```

Example 2–1 Cont'd on next page

Example 2–1 (Cont.) Running the ^CONFIG Utility

```
Answer YES to this question to start VAX DSM and make it immediately
available to all application users.

Reference: Chapter 3
```

The configuration is now started, and VAX DSM displays the following:

```
Starting VAX DSM configuration PAYROLL

Mounting volume set EMP with 1 volume
Mounting DISK$USER:[DSMMGR.DATABASE]EMPVOL1.GLS as volume 1 with 8 maps

1 Volume Set mounted

DSM Write Demon ...       Started as DSM_DEMON_1  (PID = 21E00B4E)
DSM Garbage Collector ... Started as DSM_GARCOL_1 (PID = 21E00A4F)

VAX DSM startup complete  -  User logins enabled
```

Starting Up and Shutting Down Configurations

Once a VAX DSM configuration is completely defined, you can interactively start it and shut it down. The following sections describe how to start up and shut down a configuration.

Starting a Configuration

To interactively start up DSM, log in to the VAX DSM environment manager's account. Then, run DSM with the /MANAGER command line qualifier and run the Configuration Startup Utility (^STU). For example:

```
$ DSM/MANAGER Return

          VAX DSM V6.0
             DSMMGR
           [Baseline]


>D ^STU Return

     DSM Configuration Startup


Enter the configuration identifier <PAYROLL> Return
  .
  .
  .
```

Note: Configurations cannot be started interactively while logged in to an alternate environment manager's account.

You can also configure your VMS system to automatically start a configuration at VMS system boot time. The SYS$STARTUP:DSM$INSTALL.COM procedure is supplied as a template for VAX DSM activities required at VMS system startup. You can customize this procedure for your site. The relevant portions for configuration startup in this file accomplish the following:

- Creates the DSM$BATCH batch queue where the VAX DSM startup procedure runs
- Submits the SYS$STARTUP:DSM$STARTUP.COM procedure to DSM$BATCH to start a configuration

To set up automatic configuration startup, locate and edit the portion of this file that starts the VMS Queue Manager and creates the DSM$BATCH queue. You can remove the comment characters (!) from the following command lines to create and start a local batch queue:

```
$ START/QUEUE/MAN
$ NODE = "''F$EDIT(F$GETSYI("SCSNODE"),"COLLAPSE,TRIM,UPCASE")'"
$ IF NODE .EQS. "" THEN QUEUE = "DSM$BATCH"
$ IF NODE .NES. "" THEN QUEUE = "DSM$BATCH_''NODE'"
$ INIT/QUEUE/BATCH/START/JOB_LIMIT=1 'QUEUE
```

In a single-node configuration, the name of the queue created is DSM$BATCH. In a VAXcluster environment, the name of the queue is in the following form:

DSM$BATCH_NNN

where:

NNN is the local node's SCSNODE name

Next, locate and edit the example commands that submit the SYS$STARTUP:DSM$STARTUP.COM procedure to start a configuration. For each configuration you want to start, copy these commands, remove the comment character (!), and edit the following fields to specify the configuration name and VAX DSM manager's account name:

```
$ SUBMIT/USER=uuu/PARAMETER=ppp/QUEUE='QUEUE SYS$STARTUP:DSM$STARTUP
```

where:

uuu is the environment name (VAX DSM manager's account name)

ppp is a two piece parameter list delimited by a comma in the form:

conf,login

where:

conf is the configuration name to start (if not supplied, the default configuration is started)

login is DISABLE to disable user logins and ENABLE to enable user logins (if not supplied, user logins are enabled on startup)

For each initialized environment, substitute the appropriate environment name for the VAX DSM manager's account, and supply the configuration name to start. Ensure that the DSM$INSTALL command procedure is called during system startup by inserting the following command in your site-specific startup file:

```
$ @SYS$STARTUP:DSM$INSTALL
```

If you do not specify a configuration name when you invoke or submit the SYS$STARTUP:DSM$STARTUP.COM procedure, the *default configuration* for the environment is started. The default configuration is designated when you run the ^CONFIG Utility.

Shutting Down a Configuration

You can interactively shut down a VAX DSM configuration by running VAX DSM with the /MANAGER command line qualifier. Then, use the ^SHUTDWN Utility to stop all DSM user processes and complete an orderly shutdown of the DSM Write Demon and Garbage Collector. The ^SHUTDWN Utility also removes routine mapped sections and deletes DSM system logical names. Configurations can be shut down by alternate manager accounts or other VMS users with the VMS SYSPRV privilege. For example:

```
$ DSM/MANAGER/ENVIRONMENT=DSMMGR Return

          VAX DSM V6.0
             DSMMGR
           [MGR,DAS]


>D ^SHUTDWN Return

            Processes in DSM Environment DSMMGR on 24-APR-1990 15:47

  PID    Process Name   State Device UCI,VOL Routine    Cmds   Refs    DIO Login
24600390 DSMMGR          LEF  RTA9   APL,DAS          228164  38377   3700 10:12
24600120 DSM_GARCOL_42   HIB                                        8 15:35
2460011F DSM_DEMON_42    HIB                                     3834 15:35

Continue with shutdown ? (Y/N) >Y Return
.
.
.
```

You can also use the SYS$MANAGER:DSM$SHUTDOWN.COM procedure to non-interactively shut down a configuration. For more information on system shutdown, see the section ''System Shutdown'' in Chapter 1.

> **Note:** Using the DCL STOP/ID command to stop VAX DSM user or system processes can cause all other users accessing the same VAX DSM volume set to hang, and can cause database degradation. Instead, use the VAX DSM ^FORCEX Utility, which runs down a VAX DSM user safely. For more information about how to use this utility, see Chapter 6.

Modifying Characteristics of a Configuration

When you initially created the VAX DSM environment manager account, the SYS$MANAGER:DSM$INIT.COM procedure set minimal account limits, quotas, and privileges necessary to run limited VAX DSM configurations. It is necessary to adjust these values as you modify your operating environment and configuration characteristics.

The following list describes the recommended technique for setting up a VAX DSM configuration that has all of the capabilities you require.

1. Use the ^CONFIG Utility to establish a new configuration, including the new options or values.
2. Use the ^MEMREQ Utility to determine which VMS SYSGEN parameters and account quotas need to be adjusted.
3. Establish any new VMS system or environment manager account resources required for the planned configuration. These resources, including account quotas for the VAX DSM environment manager account and parameters set during VMS SYSGEN, are described in Chapter 4.
4. Run AUTOGEN with the reboot option if you have modified any SYSGEN parameters.
5. Run VAX DSM again, and start the configuration using the ^STU Utility.

Treat configuration building as an iterative process; establish new capabilities or resources a few at a time, test the new configuration, and then add new capabilities. This approach ensures that you understand and control the effects of the changes that you make to your VAX DSM configuration.

Running Multiple VAX DSM Configurations

You can run multiple VAX DSM configurations simultaneously on a VMS system: one configuration for each VAX DSM environment. To set up a configuration for an environment, initialize a VAX DSM environment manager's account for each environment by logging in to a privileged VMS system account and running the SYS$MANAGER:DSM$INIT.COM procedure. This procedure is described in Chapter 1.

Each time you initialize a new VAX DSM environment, the manager account is given a unique *environment identifier* and *environment name*.

The environment identifier is an integer number that is used to form VMS resource names that are associated with the environment and used internally by VAX DSM.

The environment name is a systemwide unique name that is always equivalent to the VAX DSM manager's account name. The environment name is used by application users and programmers to access a configuration running in an environment.

When you define and start a configuration in an environment, two system logical names are created:

- DSM$ID_envnm

  The symbol, envnm, represents an environment name. Configuration startup equates the DSM$ID_envnm system logical name to the unique environment identifier. **This value should never be redefined.**

- DSM$DEFAULT_envnm

  The symbol, envnm, represents an environment name. DSM$DEFAULT_envnm holds the default command line qualifiers for the environment specified by envnm. The default command line qualifiers are derived from the values you selected while running the ^CONFIG Utility. The defaults specify information such as the login UCI and volume set as well as default source buffer and symbol table sizes. You can always override the defaults by appending explicit values when you enter the DSM command line.

The /ENVIRONMENT=envnm command line qualifier is used to select the environment that you want to access. For example, any VMS user can gain access to the running configuration in the environment, PRODUCTION, by entering the following DSM command line:

```
$ DSM/ENVIRONMENT=PRODUCTION Return

                VAX DSM V6.0
                 PRODUCTION
                 [WRK,VBA]

>
```

Note that, the environment selected is displayed in the VAX DSM banner along with the login UCI and volume set.

When the VAX DSM manager for environment PRODUCTION logs in to VAX DSM without the /ENVIRONMENT=envnm command line qualifier, VAX DSM attempts to access the current configuration running from the manager's account environment.

```
$ DSM Return

                VAX DSM V6.0
                 PRODUCTION
                 [WRK,VBA]

>
```

You can override this action by defining the DSM$ENVIRONMENT logical name to be equivalent to a default environment name. When you omit the /ENVIRONMENT=envnm command line qualifier, VAX DSM attempts to translate DSM$ENVIRONMENT from the process, group, or system logical name tables to determine the target environment. If the name exists, the equivalence of DSM$ENVIRONMENT is used for selecting the environment. If DSM$ENVIRONMENT is not defined, VAX DSM attempts to log you in to the configuration running in your environment.

If you access multiple environments from your account, you can define the DSM$DEFAULT_envnm logical name to select login defaults that are specific to the configurations running in each of the environments.

For example, consider a situation in which two environments named PRODUCTION and DEVELOPMENT run separate configurations concurrently on the same VMS system. You can set up different defaults for selecting the UCI and volume set to log in to when each environment is accessed. To define the defaults to access UCI WRK and volume set PRD in the PRODUCTION environment, enter the following:

```
$ DEFINE/PROCESS DSM$DEFAULT_PRODUCTION "/UCI=WRK/VOL=PRD" Return
$ DSM/ENVIRONMENT=PRODUCTION Return

                VAX DSM V6.0
                 PRODUCTION
                  [WRK,PRD]

>
```

To define the defaults to access UCI ARD and volume set DEV for the DEVELOPMENT environment, enter the following:

```
$ DEFINE/PROCESS DSM$DEFAULT_DEVELOPMENT "/UCI=ARD/VOL=DEV" Return
$ DSM/ENVIRONMENT=DEVELOPMENT Return

                VAX DSM V6.0
                 DEVELOPMENT
                   [ARD,DEV]

>
```

Initializing VAX DSM in a VAXcluster Environment

Each VAX DSM environment can extend over several processors joined in a VAXcluster configuration.

The process of initializing a VAX DSM manager account for use within a VAXcluster environment is similar to that used for a single-node system. You use the SYS$MANAGER:DSM$INIT.COM procedure to initialize a VAX DSM environment for clusterwide usage. You can run the DSM$INIT procedure while logged in to a privileged account on any node within the cluster.

DSM$INIT creates the environment directories and applies the default protections as described in the section ''Environment Directories, Files, and Protection.'' In addition, a separate directory is created for each cluster node as a subdirectory of the environment root directory.

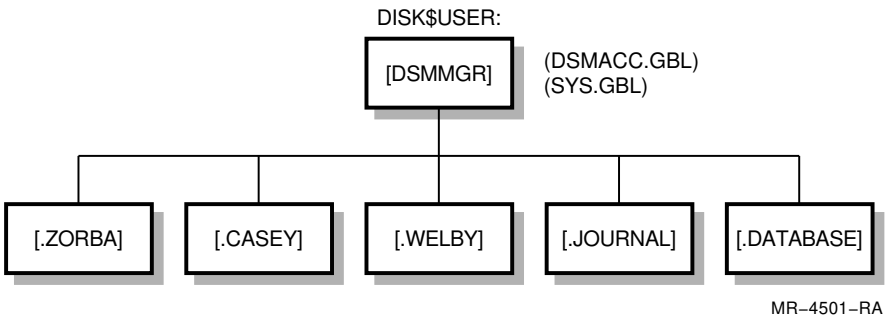
Once the manager account is initialized, VAX DSM configurations can be defined and started on each VAXcluster node. A common set of VAX DSM database sets can be created and shared simultaneously on all nodes.

Directory Structure in a VAXcluster Environment

In a single-node configuration, the VAX DSM environment root directory is used to hold log files created by the DSM system processes such as the Write Demon and Garbage Collector. In a VAXcluster environment, each node separately runs a VAX DSM configuration. The cluster node subdirectories are used to hold the log and error files generated by the DSM processes running on each of the nodes. In a cluster, the environment root directory still holds ^SYS, the configuration global, and ^DSMACC, the configuration access global, as well as routine map files. Placement of these files in the root directory allows them to be commonly accessible from all nodes in the cluster.

Figure 2–2 shows an example of the VAXcluster directory hierarchy. The directories created by DSM$INIT for the account VAXDSM are located on a cluster consisting of nodes ZORBA, CASEY, and WELBY.

Figure 2–2 Directory Hierarchy in a VAXcluster Environment



In Figure 2–2:

- The DISK$USER:[DSMMGR.ZORBA] directory holds the error and log files for DSM processes running on VAXcluster node ZORBA.
- The DISK$USER:[DSMMGR.CASEY] directory holds the error and log files for DSM processes running on node CASEY.
- The DISK$USER:[DSMMGR.WELBY] directory holds the error and log files for DSM processes running on node WELBY.

- The environment root directory, DISK$USER:[DSMMGR], holds routine map files and any RMS globals or routine directories used by all three nodes.
- The root directory contains the RMS global, SYS.GBL, which contains the configuration definitions created (using ^CONFIG) for all nodes of the cluster.
- The VAX DSM security global, DSMACC.GBL, holds user access information for all nodes of the cluster and also resides in the root directory.
- The DISK$USER:[DSMMGR.JOURNAL] directory is the default storage location for After-Image Journal or Before-Image Journal files.
- The DISK$USER:[DSMMGR.DATABASE] directory is used as the default storage location for VAX DSM volume sets.

When you run DSM$INIT in a VAXcluster environment, the following question appears so DSM$INIT can correctly define the appropriate subdirectories:

```
.
.
.

  Use of VAX DSM in a VAXcluster environment requires
  the creation of a separate subdirectory for each cluster
  member. The subdirectories hold node-specific log files
  created by VAX DSM configurations running on each of the
  VAXcluster nodes. Appropriate subdirectories will be created
  as part of this initialization process. In order to do so,
  all cluster nodes must be identified at this time.


The following VAXcluster nodes are currently booted and will be
initialized for use with VAX DSM:

CASEY   ZORBA

Are there any additional Cluster members not currently
booted that you would like to add to this list [Y OR N] ?<N> Return
.
.
.
```

DSM$INIT attempts to identify all nodes configured. However, nodes that are not booted at the time you run DSM$INIT do not appear in the list of recognized nodes. If you need to identify additional nodes, answer YES to the last question in the previous example and then supply the additional node names.

> **Note:** If you add nodes to your cluster, you have to run DSM$INIT again for all defined VAX DSM manager accounts. This ensures that the proper subdirectories are created for the new node, and then updates the appropriate DSM management globals.

Defining VAXcluster Configurations

If you are running VAX DSM in a VAXcluster environment, **you must create a separate configuration for each VAXcluster node that is going to use VAX DSM.** However, the configurations can mount common VAX DSM database sets so that DSM volume sets are accessible and shared across the cluster.

When you mount a VAX DSM database set for clusterwide access, the ^CONFIG Utility asks you whether the configuration is going to *master* the database set. You can designate one or more cluster members to master a database set. Database set masters provide DDP and After-Image Journaling services to all nodes within and outside of the cluster.

Only one master actively serves the database set at any time. However, when a database set master is shut down or fails, a database set master on another cluster node becomes the active master. For these reasons, configure more than one node as a database set master for each of the database sets that are included. While the overhead required to actively master a database set is not substantial, choose cluster members with the most available system resources.

DDP must also be enabled when you include clusterwide database sets. **When configuring DDP, you must select a unique 3-character DDP node name for each running configuration.**

See the *VAX DSM Database Operations Guide* for information about how to use DDP in a VAXcluster environment. The *VAX DSM Database Operations Guide* contains information about how to use and optimize cluster-mounted volume sets. It also contains information on how VAX DSM Journaling is implemented in a VAXcluster environment. Become familiar with the information in the *VMS VAXcluster Manual* before you attempt to run VAX DSM in a VAXcluster environment.

VAX DSM Detached Processes

This section describes the various DSM system processes that are created when you start up a VAX DSM configuration. The following VMS detached processes are created to support the environment's capabilities:

- Write Demon (writes global data modifications into the database)
- Garbage Collector (reclaims disk space freed as a result of a global KILL)
- Journal Process (records all SETs and KILLs, manages DSM transactions)
- Recovery Process (initiates an automatic recovery of the database)

- One or more DDP servers (directs communication to and from other DSM systems, and allows access to remote system globals)
- One or more DDP DECnet™ link servers (controls DECnet links to remote VAX DSM configurations)

The Write Demon and Garbage Collector are always started when you start up a configuration. The Recovery Process is started only when Before-Image Journaling is enabled for a database set. The Journal Process is started when After-Image Journaling is selected for a database set included in the configuration. If DDP is enabled, DDP servers and DDP link servers can be defined to start during configuration startup or interactively at a later time using the ^DDP Utility.

The following sections describe these detached processes in more detail. Also, for additional information about these processes, see the *VAX DSM Database Operations Guide*.

Write Demon

The Write Demon is a VMS process that must be started before any DSM user can access globals or routines residing on VAX DSM volume sets. The Write Demon creates and initializes a virtual memory section to the DSM shared data, including the disk buffers to hold global data for deferred writing to volumes. It then performs all writes to the volume using the data accumulated in the disk buffers.

When the Write Demon starts up, VAX DSM creates a log file called DSMDEMON.LOG in the log file directory that was created when the SYS$MANAGER:DSM$INIT.COM procedure was run. This file receives any errors generated by the Write Demon. Consult DSMDEMON.LOG if DSM startup fails because of a Write Demon initialization error.

If the configuration being started contains any cluster-mounted volume sets, or if the configuration contains any volume sets for which Before-Image Journaling is active, this process resolves cross-node contention for common data blocks, and controls access to the before-image journal file for each volume set.

You can print or type the Write Demon log file while the Write Demon is running. The Write Demon logs occurrences of unrecoverable disk write errors and displays them on the operator console device.

Garbage Collector

The Garbage Collector is a VMS process that the VAX DSM manager starts up when establishing VAX DSM volume sets. This process handles the asynchronous task of making blocks containing deleted data available for reallocation.

When the Garbage Collector starts up, VAX DSM creates a log file called DSMGARCOL.LOG in the log file directory that was created when the SYS$MANAGER:DSM$INIT.COM procedure was run. This file records error information about the Garbage Collector process.

You can print or type the Garbage Collector's log file while the Garbage Collector is running.

Journal Process

After-Image Journaling provides a way to keep a record of the SET and KILL commands that a VAX DSM application performs on the global database. The VAX DSM manager enables journaling for an application by selecting the appropriate startup options in ^CONFIG, and starts up the configuration. Journaling can then be enabled on a global-by-global basis.

The VAX DSM manager can interact with the Journal Process during a DSM session to control Journaling operations. To do this, the VAX DSM manager uses the Journal Process Utilities described in the *VAX DSM Database Operations Guide*.

Recovery Process

The Recovery Process is started whenever you have a database set in your configuration that has Before-Image Journaling enabled. The Recovery Process performs automatic recovery whenever VAX DSM restarts after a system or VAXcluster node failure. If VAX DSM transaction database sets are configured, the Recovery Process is also responsible for recovery from a transaction abort.

Note that in some configurations, the Recovery Process can be started to perform a single recovery operation and can then be terminated.

DDP Servers

A DDP server is a VMS process that manages the communications link between your DSM application and other DSM systems or applications. The VAX DSM manager can select the number of DDP servers to start.

DDP DECnet Link Servers

The DDP DECnet link servers are responsible for establishing DECnet connections to remote VAX DSM configurations. For DECnet DDP operations, DDP DECnet link servers transmit and receive DDP messages to the remote configuration across the DECnet circuit for all local DDP users.

One DDP DECnet link server is created for each DECnet DDP link configured in ^CONFIG.

Chapter 3
Controlling Access to VAX DSM

This chapter provides information about the security capabilities available in VAX DSM. It provides an overview of the different levels of access to VAX DSM environments and a detailed discussion of how you can control access to VAX DSM and your application. It also describes how to create application and programmer accounts.

Controlling User Access

VAX DSM can allow any VMS user to directly access a configuration running in a VAX DSM environment on the local VMS system using the /ENVIRONMENT=envnm command line qualifier. This capability, along with the ability to run multiple configurations concurrently, provides a great deal of flexibility when designing an application system. However, the VAX DSM environment manager must also restrict user's access to various components of the application system to protect the database from unauthorized access.

The modes of access as well as the access restrictions required vary greatly from one installation to another. Two models of application environments are common in MUMPS systems:

- *Dedicated application systems*

  The dedicated application system views the underlying operating system solely as a support platform with a singular purpose of serving the MUMPS application. The application designer uses the components supplied by the system to construct a turnkey system. Access to the system is controlled from the point at which the user presses Return to the point at which the application program gets control.

In this model, the use of captive terminals and captive accounts totally secures access to VAX DSM. It is likely that all application users actually use the same VMS account to access the application. The application program holds the responsibility for granting access to the database.

Very little access control is usually placed on a programmer account. The programmer is usually granted a separate VMS account with access to all UCIs, volume sets, and configurations.

This approach presents only two levels of access authority: privileged programmers and unprivileged users. It is an adequate approach in controlled operational environments that do not require any direct user interaction with the VMS system.

- *General purpose timesharing systems*

  Timesharing systems use the underlying operating system for a large variety of MUMPS and non-MUMPS services. In this model, most users have their own VMS account with direct access to the DCL command prompt ($). The MUMPS application system may be only one of the resources that is used. The user community is comprised of both privileged programmers and unprivileged users.

  In this environment, the MUMPS application system still retains its control over how the application user accesses the database once the user is running VAX DSM. However, the VAX DSM manager must be able to precisely identify which VMS users can log in to a running configuration, what modes of access are allowed, and which UCIs, volume sets, and routines are accessible. In addition, VMS files used by VAX DSM for storing database and journal information have to be protected from unauthorized modification.

  Solutions to controlling access in timesharing systems must provide many levels of access authority that are VMS account based.

A number of techniques can be used to support security for both of these application models. The selection of the most appropriate configuration of VAX DSM security features are determined by your site requirements. Regardless of the techniques you choose, the security of a DSM database ultimately depends on controlling access at the following levels:

- *VMS access control*

  Authorization to log in to a VMS system as a user

- *Environment access control*

  Authorization to read or write to database volume set files, journal files, and other VAX DSM environment files such as DSMACC.GBL and SYS.GBL

- *Configuration access control*

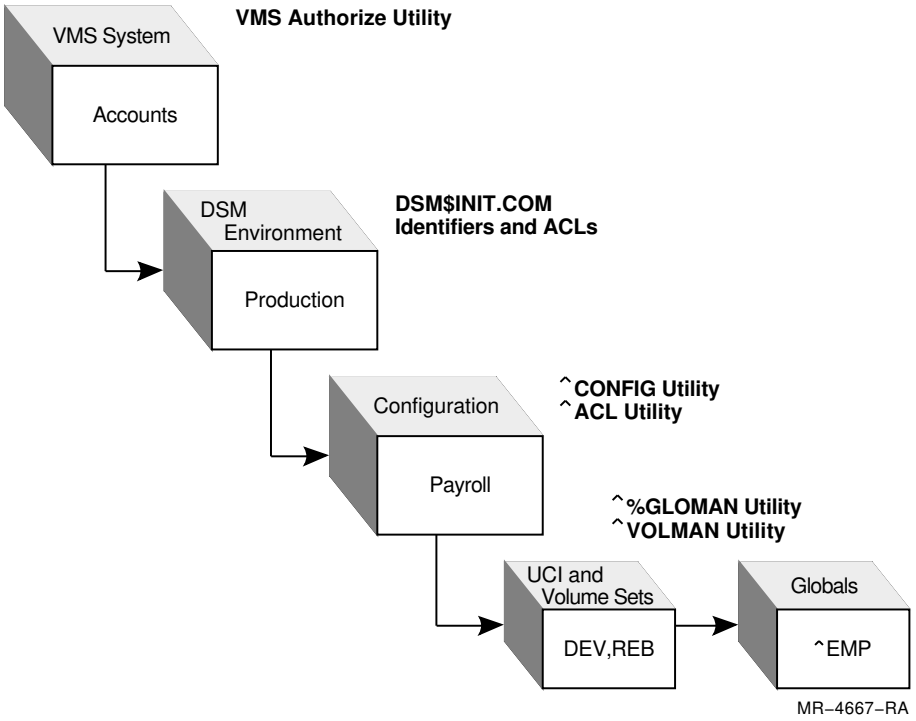
  Authorization to log in to a UCI and volume set within a running configuration

- *Global access control*

  Authorization to read or modify specific DSM globals from the login UCI

Figure 3–1 describes the relationship between the different levels of access and some of the utilities that you can use to provide access control.

Figure 3–1 Access Control Relationships



The following sections describe configuration access control methods available in VAX DSM. They describe the utilities provided by VAX DSM as well as system conventions you can use to construct access methods for either dedicated application or timeshare systems.

For information about global access control, see the *VAX DSM Database Operations Guide*. It provides detailed information about the protection levels available for DSM globals and the utilities you can use to establish protection on a global by global basis.

The *Guide to VMS System Security* contains a useful overview of evaluating the security needs of a site. Refer to this manual for detailed information about VMS system security.

VMS Access Control

VMS user *accounts* and *passwords* allow the VMS system manager to establish which individuals can log in to a VMS system and the various rights and privileges the user is granted. Users with sufficient VMS privileges can perform system management functions and access any file on the system.

You can create different types of VMS accounts for different purposes:

- Less privileged accounts can be created to allow access to the DCL command line interpreter so that the user can run programs and access private files (as well as unprotected public files).
- *Captive accounts* can be configured to allow users to log in to VMS with or without a required password. However, the captive users are tied to a login command procedure and are not allowed to execute any general functions outside of this procedure. These accounts serve as dedicated application accounts and generally exist for the singular purpose of running a specific application.

In a MUMPS environment, captive accounts can be used to tie application users directly to a VAX DSM application. Unprivileged programming accounts can be created to allow a VAX DSM application developer to access DSM from the DCL command line. The programmers can select a number of DSM log in options, UCIs or volume sets, but they would not have sufficient privilege to access other VMS files or protected services.

For complete information about creating VMS accounts, see the *Guide to Setting Up a VMS System*.

Environment Access Control

Directories, files, and management functions are *objects* within a VAX DSM environment that must be protected from unauthorized access. Files, such as the database volumes or after-image journal files are subject to inspection by VMS users through the use of general purpose VMS utilities and system calls. The DCL COPY or DELETE commands can be used to manipulate these files. The VMS Dump Utility can be used to display the contents of volume sets.

Protecting Environment Files

The VMS operating system offers two primary mechanisms for use in protecting access to objects such as files:

- *UIC-based protection*

  This protection is based on the user identification code (UIC) of the manager account and is applied to all user owned files. Access privileges are controlled according to SYSTEM, OWNER, GROUP and WORLD categories. READ, WRITE, EXECUTE, and DELETE access can be specified for each category.

- VMS *Identifiers* and *access control lists* (ACLs)

  VMS identifiers are records defined in the system rights database, SYS$SYSTEM:RIGHTSLIST.DAT. System managers create and grant identifiers to specific VMS accounts using the VMS Authorize Utility. When you grant an identifier to an account, the account is considered a *holder* of the identifier.

  ACLs are access control structures that can be associated with an object such as a file. ACLs consist of a list of entries, known as *access control entries* or *ACEs*. Several types of ACEs exist. An *identifier ACE* describes the access rights granted or denied to holders of the identifier.

  You can control which users can access an object by creating an identifier and then granting it to a user. An ACL that includes the appropriate identifier ACE is then applied to the object. All holders of the identifier can then access the object.

When you initialize a new VAX DSM environment, the SYS$MANAGER:DSM$INIT.COM procedure creates an unprivileged VMS account that is used to manage VAX DSM operations within the environment. UIC-based ownership protections are applied to all directories and files created. The UIC protection scheme is:

```
SYSTEM: READ, WRITE, EXECUTE
OWNER:  READ, WRITE, EXECUTE
GROUP:  NO ACCESS
WORLD:  NO ACCESS
```

This scheme allows only VMS system managers and users with the same VMS UIC access to VAX DSM environment files.

A VMS identifier that is unique to the environment is also created in the rights database during environment initialization. The identifier is placed as an entry in ACLs that are associated with each of the files in the environment. This scheme allows holders of the identifier to access files within a VAX DSM environment.

The VAX DSM environment manager account is always granted the manager identifier. The manager can use the VAX DSM Access Control Utility (^ACL) to grant the identifier to other VMS users and designate them as *alternate environment managers*.

The name of the identifier is in the form:

DSM$MANAGER_envname

where:

envname is the name of the environment manager account

The ACL placed on each of the environment directories also contains an access control entry known as a *default protection ACE*. The placement of the default protection ACE along with the identifier ACE causes the proper UIC- and identifier-based protections to be propagated to all files created within the directories. This ensures that newly created volume set and journal files receive proper access protection.

Maintaining Environment Security

The security of a VAX DSM environment depends on ensuring that all of the files and directories within the environment are properly protected. Use the following guidelines to ensure that the protections are sufficient:

- Create all new volume sets and journal files within directories created by DSM$INIT.
- Inspect the protection of newly created files using the DCL DIRECTORY/SECURITY command.
- Apply appropriate ACLs and UIC-based protections on all directories or files when necessary.

The DSM$INIT command procedure applies appropriate protections to all directories and files in the environment. Directory protections are propagated to files created within the directories. It is a good idea to always create and locate new volume set files and journal files within properly initialized directories.

If you move or restore files to an environment directory, inspect the applied protections to ensure they are correct. The security placed on a properly configured environment root directory for the DSMMGR environment is shown in the following example:

```
$ DIRECTORY/SECURITY DISK$USER:[0,0]DSMMGR.DIR Return

DSMMGR.DIR;1        [DSMMGR]       (RWE,RWE,,)
     (DEFAULT_PROTECTION,SYSTEM:RWED,OWNER:RWED,GROUP:,WORLD:)
     (IDENTIFIER=DSM$MANAGER_DSMMGR,OPTIONS=DEFAULT,ACCESS=READ+WRITE+
      EXECUTE+DELETE)
     (IDENTIFIER=DSM$MANAGER_DSMMGR,ACCESS=READ+WRITE+EXECUTE)
```

The security placed on a properly configured environment file for the environment DSMMGR is shown in the following example:

```
$ DIRECTORY/SECURITY DISK$USER:[DSMMGR]SYS.GBL Return

Directory DISK$USER:[DSMMGR]

SYS.GBL;1             [DSMMGR]      (RWE,RWE,,)
    (IDENTIFIER=DSM$MANAGER_DSMMGR,ACCESS=READ+WRITE+EXECUTE)
```

Note that some DCL operations, such as the DCL RENAME command, may not apply directory level protections when a file is moved into the directory. In these cases, you can apply the correct UIC and identifier protections. The following example shows the commands used to establish the correct file protection for the DSMMGR environment in UIC [250,1]:

```
$ RENAME DISK$DATA:[DATABASE]AAAVOL1.GLS -Return
DISK$DATA:[DSMMGR.DATABASE]AAAVOL1.GLS Return

$ SET ACL/LIKE=(OBJECT_NAME=DISK$USER:[DSMMGR]SYS.GBL) -Return
DISK$DATA:[DSMMGR.DATABASE]AAAVOL1.GLS Return

$ SET FILE/PROTECTION=(O:RWE,S:RWE,G,W) -Return
DISK$DATA:[DSMMGR.DATABASE]AAAVOL1.GLS/OWNER=[250,1] Return
```

Configuration Access Control

Access to a running configuration must be controlled to prevent unauthorized VMS users from running VAX DSM management procedures or accessing protected globals. The objective of configuration access control is to restrict a user to specific *access modes*, UCIs, and volume sets.

VAX DSM Access Modes

VAX DSM provides three modes of access when logging in to a configuration. Specific privileges are associated with each of the following access modes:

- *Manager Mode*
- *Programmer Mode*
- *Application Mode*

The ^CONFIG Utility allows you to optionally enable the VAX DSM configuration access control system. If you do not enable the access control system, any VMS user is allowed access to the configuration in Programmer Mode or Application Mode without restriction. Manager Mode access is always protected.

If you enable the access control system, you can use the ^ACL Utility to grant Programmer Mode or Application Mode access to individual VMS accounts. Programmer Mode access can be restricted to specific UCIs. Application Mode access can be restricted to specific UCIs and application routines.

The access control system gives you complete control over access to a VAX DSM configuration. You can allow selected VMS users to access VAX DSM in a controlled fashion. Access is controlled from the DCL DSM command. Consequently, it is not necessary to set up captive application accounts or programmer accounts to limit user access to VMS. The access control system is particularly suitable for a general purpose timesharing environment where the entire community of VMS users are potential VAX DSM users.

The following sections describe the access modes and their associated privileges in more detail.

Manager Mode Access

You enter Manager Mode by running VAX DSM with the /MANAGER command line qualifier:

```
$ DSM/MANAGER Return

          VAX DSM V6.0
             DSMMGR
           [MGR,AAA]
```

Manager Mode allows you to interact directly with the VAX DSM interpreter for the purposes of creating volume sets, configurations and other VAX DSM management functions. When you run VAX DSM in Manager Mode, VAX DSM enforces the following default options, regardless of any other options specified on the command line:

- UCI 1 on volume set 0 is used for default globals and routine access
- /NOPACK/NOPASSALL is enforced
- /NOMAPPED — mapping to routines in virtual memory sections is disabled
- /NOTRANSACTION — transaction recovery unit processing is disabled
- A minimum value of interpreter defaults are used for the following qualifiers:

  /SYMBOL_TABLE_SIZE=102080
  /STACK_SIZE=8192
  /SOURCE_BUFFER_SIZE=16384

VAX DSM associates specific privileges to users who run VAX DSM with the /MANAGER command line qualifier. Manager Mode gives you access to:

- VAX DSM management utilities in the MGR UCI
- Unrestricted use of the VIEW command and $VIEW function
- Privileged system supplied $ZCALLs including %SETUCI
- DSM /BYPASS command line qualifier

Manager Mode access is available to an account only if the account is:

- Initialized as the environment manager account using the DSM$INIT procedure
- Designated as an alternate environment manager account using the VAX DSM Access Control Utility (^ACL)
- A privileged VMS account with the SYSPRV or BYPASS privileges

Note: The restrictions to Manager Mode access apply even if you use ^CONFIG to disable the configuration access control system.

Programmer Mode Access

You enter Programmer Mode by running VAX DSM without specifying a routine reference on the DSM command line:

```
$ DSM/UCI=PHA/ENVIRONMENT=PERSONNEL Return

          VAX DSM V6.0
           PERSONNEL
           [PHA,EMP]
```

Programmer Mode allows you to interact directly with the VAX DSM interpreter. The privileges associated with Programmer Mode include:

- Create, store, and run MUMPS routines interactively.
- Set or kill any global data accessible from the current UCI.
- Run VAX DSM Library Utility routines.
- Access VAX DSM library $ZCALL functions directly.

If you enable the configuration access control system, you use the ^ACL Utility to grant VMS users access to specific UCIs and volume sets. When the access control system is enabled, Programmer Mode users do not have access to the following capabilities:

- VIEW command and $VIEW function
- %SETUCI $ZCALL library function to non-authorized UCIs
- DSM /BYPASS command line qualifier
- MGR UCI
- VAX DSM privileged $ZCALLs (DSM ZCALL package)
- Editing Library (%) Utilities

VAX DSM always allows the use of these capabilities while running a library (%) utility.

Application Mode Access

You enter Application Mode by specifying a routine name on the DSM command line. For example:

```
$ DSM/UCI=APP/ENVIRONMENT=PERSONNEL ^APPL Return
```

In Application Mode, the user is tied directly to the application routine that is referenced on the command line. The ability to read or modify the DSM database or change the current volume set or UCI is controlled by the predefined DSM routines. Therefore, the user's access to sensitive data is completely defined by the application programs.

If you enable the configuration access control system, you use the ^ACL Utility to grant VMS users access to specific UCIs, volume sets, and routines. Application users who do not also have Programmer Mode access to the login UCI are not allowed to escape from the application routine into Programmer Mode using the ZESCAPE command.

Global Access Control

VAX DSM uses a protection scheme that defines global access control based on a user's default (login) UCI name and volume set. The four classes of users are:

WORLD	Users logged in to any UCI on any volume set
GROUP	Users logged in to a UCI with a name that begins with the same letter as the UCI name of the target global
SYSTEM	Users logged in to the 1st UCI on volume set 0 (MGR)
UCI	Users logged in to the UCI that holds the target global

A separate protection level for each of these four classes of users can be applied to a global. The three protection levels defined are:

Read	Global get operations
Write	Global set operations
Delete	Global kill operations

These protection levels can be applied to each class of user for each global in any combination.

The ultimate objective of all of the protection mechanisms described in this section is to control the access to globals in the VAX DSM database. The application of appropriate global protections determines which users gain access to a global while logged in to a UCI and volume set. Login access to UCIs and volume sets is controlled by assigning VMS account privileges, selecting configuration access control methods, and granting programmer or application user access to a specific UCI.

For a complete description of VAX DSM global protection, see the *VAX DSM Database Operations Guide*.

Establishing VMS Accounts for Programmers and Application Users

Each user accessing a VMS system has an account. A user can be a DSM programmer or an application user. The VMS system's user authorization file, SYS$SYSTEM:SYSUAF.DAT, contains information about each VMS user account. The VMS operating system uses this information to identify valid user accounts. For complete information on how to create VMS accounts, see the *Introduction to VMS System Management*.

The VAX DSM environment manager is responsible for establishing programmer accounts and application user accounts and ensuring the security of these accounts. To create a VMS account, you must have READ/WRITE access to the SYSUAF.DAT file.

The first step in setting up an account is to determine what you want to establish as the account's user name, password, default directory, and other identifying information.

The next step is to create the account's main directory and (if appropriate) a login command file. Finally, enter information about the account into the system's user authorization file (UAF) using the VMS Authorize Utility. Use the VMS Authorize Utility to specify the user name, password, default device, and other identifying information.

The following example shows how to use the VMS Authorize Utility to create a programmer or application user account for the VAX DSM configuration with a UIC of [215,7].

```
$ SET DEFAULT SYS$SYSTEM Return
$ RUN AUTHORIZE Return
UAF>ADD TILLMAN/PASSWORD=GREMLINS/UIC=[215,8]- Return
/DEVICE=DISK$USER:/DIRECTORY=[TILLMAN]- Return
/BYTLM=40960/ENQLM=300/WSQUOTA=1000/WSEXTENT=1200/FILLM=150- Return
/PRIV=TMPMBX/DEFPRIV=TMPMBX Return
```

The *Guide to Setting Up a VMS System* contains instructions for using the SYS$EXAMPLES:ADDUSER.COM procedure for adding a new user to the UAF. This command procedure prompts for all categories of information about the user, runs the VMS Authorize Utility, and cleans up after itself. You may find that if you copy and use this command procedure, it is more convenient than performing all of the steps yourself.

VAX DSM users need larger quotas in some areas than the ordinary VMS user. Make sure that you establish appropriate values for the BYTLM, ENQLM, WSQUOTA, FILLM, and WSEXTENT quotas, as described in the section ''Process Quotas for VAX DSM Users'' in Chapter 4. VAX DSM does not run unless these limits are adequate.

VAX DSM programmers must have specific privileges to perform some VAX DSM operations as described in Table 3–1.

Table 3–1 VMS System Privileges Needed by VAX DSM Programmers

DSM Command	Description	Privilege
ZUSE	$BRKTHRU to a terminal	OPER
JOB	Starts a DSM process	TMPMBX
OPEN x:(MAILBOX:NEWVERSION)	Creates a temporary mailbox	TMPMBX
OPEN x:(MAILBOX:NEWVERSION:SYSTEM)	Creates a permanent mailbox	PRMMBX SYSNAM

The %GETUAI $ZCALL requires VMS privileges. For more information on using the %GETUAI $ZCALL and on creating mailboxes, see the *VAX DSM Programmer's Guide*.

Configuration Access Control Utilities

If you enable configuration access control, Programmer Mode access can be granted to a specific list of volume sets and UCIs. The user can only run VAX DSM in Programmer Mode in those volume set and UCI combinations to which access was granted. Once running DSM, the user can use the %SETUCI $ZCALL to move to only those volume sets and UCIs for which login is authorized. The Programmer Mode user has access to all utilities in the library directory (routines that begin with %). Programmer Mode access to a volume set and UCI includes Application Mode access to all of its routines.

Application Mode access is also granted to a specific list of volume sets and UCIs. In addition, login is restricted to specified routines within the volume set and UCI combination. Users limited to Application Mode access can only log in to a DSM configuration with the following command line:

```
$ DSM[/qualifier...] ^ROUTINENAME
```

> **Note:** You cannot specify a label entry point when using the ^ACL Utility to restrict access to an application.

Users do not escape into Programmer Mode if the program they are running encounters the ZESCAPE command unless they also have Programmer Mode access to the current volume set and UCI.

Granting Access Interactively (^ACL)

The ^ACL Utility defines account access rights lists for any configuration started up in the DSM environment from which it is run.

Enter the following to run the ^ACL Utility:

```
$ DSM/MANAGER Return

          VAX DSM V6.0
             DSMMGR
           [MGR,AAA]

>D ^ACL Return
```

The utility displays the following menu:

```
Environment Access Utilities

    1.  ADD/MODIFY USER                  (ADD^ACL)
    2.  DELETE USER                      (DELETE^ACL)
    3.  MODIFY ACTIVE AUTHORIZATIONS     (^ACLSET)
    4.  PRINT                            (PRINT^ACL)

Select option >
```

You can get help at any question by entering a question mark (?); you can get a listing of current values by entering ^L; and you can return to the previous question by entering a circumflex (^). All responses, except the user name, can contain trailing wildcard characters. Any entry can be deleted by prefixing a minus sign (–) to the entry. For example, you can remove access to UCI RAD by entering –RAD at the UCI prompt.

The ADD/MODIFY USER option (ADD^ACL) allows you to edit the DSM access rights for a VMS account. The following is an example of the ADD^ACL Utility:

```
>D ADD^ACL Return

VMS User Name:   >TILLMAN Return

ACCESS MODE    VOL       UCI       ROUTINE
-----------    ---       ---       -------

No access rights for this user.

Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):  >A Return
Volume set name:   >LAB Return
UCI:   >PHA Return
Routine:   >RX Return
Routine:   > Return
Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   >A Return
Volume set name:   <LAB> Return
UCI:   <PHA>CHM Return
Routine:   >ORDER Return
Routine:   >INQUIRE Return
Routine:   > Return
Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   >A Return
Volume set name:   <LAB> Return
UCI:   <CHM>RAD Return
ROUTINE:   >INQUIRE Return
ROUTINE:   > Return
Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   > Return
```

```
USER            ACCESS MODE     VOL        UCI        ROUTINE
----            -----------     ---        ---        -------

TILLMAN         APPLICATION     LAB        CHM        INQUIRE
                                                      ORDER

                                           PHA        RX

                                           RAD        INQUIRE

OK to file?   <Y> Return

VMS User Name:    > Return

OK to activate changes now?  <Y> Return
```

Note: When running a configuration in a VAXcluster environment, you must run the ^ACLSET Utility on each node to activate modifications across the cluster.

In the previous example, a new user named TILLMAN is granted user (Application Mode) access to volume set LAB. TILLMAN is able to run program RX in the PHA UCI, programs ORDER and INQUIRE in the CHM UCI, and program INQUIRE in the RAD UCI. After you enter the user name, the program lists the user's current access rights (in this case none). Each prompt shows the context of the current entry. For example, the user name, access mode, and volume set for which you are being asked to grant access to a UCI. At the end of the session, you are shown the rights list before you are asked to file it. You can then enter information on another user or press Return. You can activate the changes immediately or answer NO to the last question. If you do not activate the changes, you must use the ^ACLSET Utility to activate those changes.

The DELETE USER option (DELETE^ACL) allows you to delete the entire rights list associated with a user name, prohibiting access by that user to a DSM environment controlled by this rights database. For example:

```
>D ADD^ACL Return

VMS User Name    >TILLMAN Return

ACCESS MODE     VOL        UCI        ROUTINE
-----------     ---        ---        -------

USER            AAA        CHM        INQ
                                      REG

                           PHA        LAB

                           RAD        INQ


OK to delete?    <Y> Return
   TILLMAN DELETED
```

The MODIFY ACTIVE AUTHORIZATIONS option (^ACLSET) takes the global created by ADD^ACL, writes it to a file, and maps it to a global section. You must run ^ACLSET if you do not answer YES to the following question at the end of ADD^ACL:

```
OK to activate changes now? <Y>
```

The PRINT option (PRINT^ACL) prints the rights lists of all users on your terminal, or writes it to a VMS file.

Granting Access Noninteractively (^ACLCALL)

The ^ACLCALL program provides an noninteractive alternate method of adding and deleting user access to a VAX DSM environment. This program can be called from another program in the manager UCI or from the DCL command line. Use of this program requires DSM Manager Mode access to the environment.

Add a user access by a call from a MUMPS program as follows:

```
D ADD^ACLCALL(USER,MODE,VOL,UCI,ROUTINE)
```

Delete a user access by a call from a MUMPS program as follows:

```
D DELETE^ACLCALL(USER,MODE,VOL,UCI,ROUTINE)
```

Add a user access from DCL as follows:

```
$ DSM/MAN/DATA="USER,MODE,VOL,UCI,ROUTINE" DCLADD^ACLCALL
```

Delete a user access from DCL as follows:

```
$ DSM/MAN/DATA="USER,MODE,VOL,UCI,ROUTINE" DCLDEL^ACLCALL
```

where:

USER	is the VMS user name
MODE	is one of MANAGER, PROGRAMMER, or APPLICATION (can be abbreviated to 1 character ("M", "P", or "A"))
VOL	is 3-character volume set name (for example, AAA or RAD or can be specified with a trailing wildcard AA* or R*)
UCI	is a 3-character UCI name (for example, AAA or RAD or can be specified with a trailing wildcard AA* or R*)
ROUTINE	is a MUMPS routine name (required when granting application mode access)

Note: The ^ACLCALL program updates the permanent access database for an environment. In order to update the active database you need to run the ^ACLSET Utility either from within MUMPS (D ^ACLSET), or from DCL ($ DSM/MAN ^ACLSET). If you are adding or deleting a number of users, run ^ACLSET after all invocations of ^ACLCALL.

Creating Dedicated Application Accounts

The VAX DSM manager can establish separate VAX DSM application accounts for any of the following reasons:

- To enable protected and dedicated startup of an application
- To provide an easy-to-use interface for the application user
- To prevent tampering with the data that is manipulated by an application

Application accounts can be set up to effectively *tie* an application user to a specific application, which prevents any interaction with the VMS operating system.

The process of establishing dedicated application accounts and terminals requires that you configure a number of components in a manner that directly connects the application user to the application programs. These components include:

- VMS automatic login facility (ALFMAINT.COM)
- VMS captive user accounts
- VMS login command procedures
- VAX DSM command line qualifiers
- VAX DSM tied terminal utilities

Two login methods are provided for application users:

- *Tied Application Accounts*

  The application user enters a user name and password that correspond to a particular VAX DSM application account. On login, a *captive* login command procedure is called directly. The DSM application is run from the login command procedure without further interaction, and logs the user out when the application session is complete.

- *Tied Terminals*

  You can tie an application account to a terminal (or set of terminals) through the VMS automatic login facility. Use the SYS$MANAGER:ALFMAINT.COM procedure to set up a terminal that accepts automatic logins from authorized users (tied terminals). For a description of this facility, see the *Guide to VMS System Security*. When you press control characters (such as Return or Ctrl/c) on a tied terminal, the login command procedure is called directly without further user interaction.

Both techniques described previously accomplish the same objective. The application user is granted access to a VMS account that you have set up for dedicated application use. The remainder of this section describes how to set up captive accounts, create login command procedures, and use the ^TIED Utility to tie accounts to an application.

Setting Up a Captive Account

You can set up a VMS captive account to provide a user account that is dedicated to running the DSM application. The VMS captive account is set up by logging in to the VMS system manager's account and using the VMS Authorize Utility. For example:

```
$ SET DEFAULT SYS$SYSTEM Return
$ RUN AUTHORIZE Return
UAF> ADD TILLMAN/UIC=[215,8]- Return
/DEVICE=DISK$USER:/DIRECTORY=[TILLMAN]- Return
/BYTLM=40960/ENQLM=300/WSQUOTA=1000/WSEXTENT=1200/FILLM=150- Return
/PRIV=TMPMBX/DEFPRIV=TMPMBX- Return
/FLAGS=(CAPTIVE,DISWELCOME,DISCTLY,DISREPORT)/LGICMD=CAPTIVE.COM/NOPASSWORD Return
```

The previous example sets the following flags and qualifiers:

- The *CAPTIVE* flag indicates that the account user cannot change any login defaults or have direct access to DCL.
- The *DISWELCOME* flag disables the VMS ''Welcome to . . . '' message.
- The *DISCTLY* flag disables Ctrl/Y.
- The DISREPORT flag disables the last login time reporting message.
- The /NOPASSWORD qualifier sets up the account so that you do not need to enter a VMS password after you enter the VMS user name.
- The /LGICMD= qualifier specifies that the CAPTIVE.COM procedure is run when the application user logs in.

The *Guide to Setting Up a VMS System* contains instructions for using the SYS$EXAMPLES:ADDUSER.COM procedure for adding a new user to the UAF. This command procedure prompts for all categories of information about the user, runs the VMS Authorize Utility, and cleans up after itself. You may find that if you copy and use this command procedure, it is more convenient than performing all of the steps yourself.

Creating a Captive Command Procedure

You can create a command procedure that runs when a user logs in to a captive account. The command procedure invokes DSM in Application Mode and runs a DSM routine. You can use the DSM Tied Terminal Utility (^TIED) to dispatch the process to the appropriate application routine, or you can invoke your own routine.

If you are using the ^TIED Utility, include the following lines in CAPTIVE.COM:

```
$ DSM/INPUT=SYS$COMMAND/ENVIRONMENT=DSMMGR/VOL=XXX/UCI=MGR ^TIEDSTU
$ LOGOUT
```

To use the ^TIEDSTU Utility, the VMS user account must be granted Application Mode access to the MGR UCI in the designated volume set. In addition, because the ^TIED Utility stores tied terminal port and routine tables in the SYS.GBL RMS global, you must allow read accessibility to SYS.GBL. To configure proper protection for the environment root directory and the SYS.GBL global, you can apply ACL entries that use the DSM$USER_envnm identifier. For example, to allow read access to SYS.GBL in the environment root directory, DISK$DATA:[DSMMGR], enter the following commands:

```
$ SET DIRECTORY DISK$DATA:[DSMMGR]/ACL= -Return
            (IDENTIFIER=DSM$USER_DSMMGR,ACCESS=EXECUTE) Return

$ SET ACL DISK$DATA:[DSMMGR]SYS.GBL/ACL = -Return
            (IDENTIFIER=DSM$USER_DSMMGR,ACCESS=READ) Return
```

Alternatively, you can use an application specific entry routine to control access by a captive account. Include the following lines in CAPTIVE.COM, substituting your environment, volume set name, UCI, and routine names:

```
$ DSM/INPUT=SYS$COMMAND/ENVIRONMENT=DSMMGR/VOL=xxx/UCI=xxx ^ROUTINE
$ LOGOUT
```

> **Note:** If you use a customized login method and the security system is enabled for the running configuration, you must also use the ^ACL Utility to grant the VMS user account Application Mode access to the specified volume set, UCI, and routine.

Setting Up VAX DSM Tied Routine and Port Tables

The Tied Routine Utility (^TIED) dispatches to a user-defined application routine from the VAX DSM manager's UCI. The user routine can be located in any UCI or volume set mounted in the currently running configuration. The routine dispatch is keyed to the name of the port that the user is logged in to by using the value of the $ZIO special variable. This facility allows the VAX DSM manager to create a table of ports and associate the port with a specific application routine.

The ^TIED Utility provides a menu interface to all tied routines and port maintenance functions described. Enter the following to run the ^TIED Utility:

```
>D ^TIED Return
```

The utility displays the following menu:

```
Tied Routine Utilities:

    1. ADD PORT                 (ADD^TTPORT)
    2. DELETE PORT              (DELETE^TTPORT)
    3. MODIFY ROUTINE TABLE     (MODIFY^TTROU)
    4. SHOW TIED PORTS          (SHOW^TTPORT)
    5. SHOW TIED ROUTINES       (SHOW^TTROU)

Enter Option >
```

Tie routines using the following procedure:

1. First, create a Tied Routine Table using the MODIFY^TTROU Utility. Each routine in the table is created and indexed by a unique number.

   ```
   Enter Option >3 Return MODIFY ROUTINE TABLE
   Add Tied Routine Table Entries

   Enter tied routine index number <1> Return

   Table entry not currently defined
   Enter UCI name for routines> REB Return
   Enter Volume Set name for routines> REB Return
   Enter routine name > ^ACCT Return
   Enter UCI name for globals <REB> Return
   Enter Volume Set name for globals <REB> Return

   ^ACCT added as tied routine number 1

   Enter tied routine index number <2> Return
   ```

2. Next, create a Tied Port Table entry for each port or group of ports you want to tie to an application routine. Do this by using the ADD^TTPORT Utility. For each port, the ADD^TTPORT Utility asks you to specify the Tied Routine index number associated with the tied routine. The organization of these two tables is designed to allow you to modify the routine name of an entry in the Tied Routine Table without having to rebuild the Tied Port Table.

   When you build the Tied Port Table, use a wild card to specify a port name to force ports of one type to be tied to one routine. For example, specifying TX* as a port to tie, forces all TX-type terminals to connect to a designated tied routine.

   ```
   Enter Option >1 Return ADD PORT

   Add Tied Ports

   Enter tied routine number > 1 Return

   Enter port names to tie to ^ACCT

   Port(s) >TXA0: Return
   Port(s) > Return

   TXA0:     tied to routine #1

   Enter tied routine number > Return
   ```

Use the DELETE PORT option (DELETE^TTPORT) to delete an entry from the table. For example:

```
Enter Option >2 Return DELETE PORT

Delete Tied Ports

Enter port names to delete

Port(s) TXA1: Return

TIED PORTS TABLE updated
PORT(s) > Return
```

Use the SHOW TIED PORTS option (SHOW^TTPORT) to display the Tied Ports Table and the SHOW TIED ROUTINES option (SHOW^TTROU) to display the Tied Routines Table entries.

Controlling Access to the Callable Routines Interface

The VAX DSM callable interface allows you to develop applications written in languages other than MUMPS to call the VAX DSM interpreter and reference the VAX DSM database. The interface is comprised of standard entry points that can be referenced by any program written by a VMS user. Consequently, to completely secure your VAX DSM database, restrictions must be placed on the use of the callable interface.

Access control for the callable routines is provided for each of two classes of calls to VAX DSM:

- DSM$DSM routine

  The DSM$DSM routine provides a DCL command line type interface to VAX DSM from external programs. Any valid DSM command line can be passed to the routine, but the command line must end with a valid MUMPS routine reference.

  Use of the DSM$DSM routine is granted depending on whether the user running the external routine has been authorized to run VAX DSM in Application Mode. If you do not enable the VAX DSM configuration access control system, authorization is not necessary. If you enable the access control system, you must use the ^ACL Utility to grant Application Mode access to the user.

- DSM$INITIALIZE routine

  The DSM$INITIALIZE routine and associated routines allow access to all global and interpreter functions. The calling program supplies a login UCI and volume set. Once DSM$INITIALIZE is complete, the calling program can set, kill, and read any MUMPS global within the UCI or data that is unprotected in the database. This functionality is similar to the capabilities available to users entering VAX DSM in Programmer Mode.

Because of these capabilities, use of the DSM$INITIALIZE routine is granted depending on whether the user running the external routine has the authorization to run VAX DSM in Programmer Mode. If you do not enable the VAX DSM configuration access control system, authorization is not necessary. If you enable the access control system, you must use the ^ACL Utility to grant Programmer Mode access to the user.

VAX DSM overrides the normal checks made to determine whether a user has the authorization to use the callable interface if:

- The user has the VMS SYSPRV or BYPASS privileges.
- The user is the VAX DSM environment manager.
- The main calling image is installed with the VMS TMPMBX privilege using the VMS Install Utility.

Installation of the calling program allows you to give unprivileged VMS users the ability to run an application program in a highly controlled fashion. The user can run the program, but cannot modify the actions of the calling program or write private programs that access VAX DSM. Installation of a program can only be done by a privileged VMS user with the VMS CMKRNL privilege:

```
$ SET PROCESS/PRIVILEGE=CMKRNL Return
$ INSTALL ADD DISK$APPL:[APPL]APPLPROG.EXE/PRIVILEGE=TMPMBX Return
```

During development of the calling program, you can grant the developer the appropriate access to the UCIs that are necessary for testing. Once the program is developed and debugged, you can install the program using the VMS Install Utility to make it available for general use.

Using the Login Control Utility (^LOGIN)

Many operational and system management functions require that no application programmers or users are currently accessing the running configuration. Functions such as backup, database integrity checking, or database repair require standalone operations.

The Login Access Control Utility (^LOGIN) allows the VAX DSM environment manager to control all user and programmer access to running VAX DSM configurations. When login access is disabled, users who attempt to run VAX DSM without the /MANAGER command line qualifier receive the following error message:

```
%DSM-E-LOGINDIS, VAX DSM logins are currently disabled
```

Authorized VAX DSM managers who run VAX DSM with the /MANAGER command line qualifier are always granted access.

User access can dynamically be disabled and reenabled through use of the ^LOGIN Utility. This allows the VAX DSM manager to disable VAX DSM access without having to disable interactive logins to the VMS operating system.

Enter the following to run ^LOGIN:

```
>D ^LOGIN Return
```

The utility displays the following options:

```
Login Control Utilities

     1. DISABLE               (DISABLE^LOGIN)
     2. ENABLE                (ENABLE^LOGIN)

Select Option >
```

Select Option 1 to disable logins and Option 2 to enable logins.

Using the ^STU Utility, you can optionally choose to have user logins disabled when configuration startup is complete. This starts VAX DSM allowing manager access only. When you are ready to allow application users back into VAX DSM, you reenable login access using the ^LOGIN Utility.

```
$ DSM/MAN Return

>D ^STU Return
     DSM Configuration Startup


Enter the configuration identifier   <WRITER>
Enable user logins on startup [Y OR N] ?   <Y>

Starting VAX DSM configuration WRITER
.
.
.
VAX DSM startup complete - User logins enabled
```

In addition, an optional input parameter can be used to disable user access on configuration startup when the SYS$MANAGER:DSM$STARTUP.COM procedure is called. The calling format for DSM$STARTUP.COM is as follows:

```
$ @SYS$STARTUP:DSM$STARTUP p1 p2
```

where:

p1 is the name of the configuration to start or "" to start the default configuration

p2 is DISABLE to disable user logins on startup and ENABLE to enable user logins (default)

Chapter 4
Tuning VMS for Use with VAX DSM

This chapter describes the relevant VMS process and system parameter adjustments necessary to run VAX DSM. It also describes how to configure shared and nonpaged memory.

Overview

Tuning the VMS operating system for use with VAX DSM involves adjusting the following:

- Process quotas for VAX DSM users
- VMS SYSGEN parameters

The first two sections describe the process quotas and SYSGEN parameters that may require modification. The section ''Using the ^MEMORY Utility'' contains information on how you can use the ^MEMREQ Utility to determine the proper values to use when you modify process and system parameters. The section ''Modifying VMS System Parameters'' describes the recommended method for modifying VMS system parameters. The section ''Page and Swap File Size'' discusses considerations for creating page and swap files.

Process Quotas for VAX DSM Users

Each VMS user (including the VAX DSM manager) must have set limits on certain system resources that affect system performance. Use the VMS Authorize Utility to modify process quotas for each user.

The following are the VMS limits that directly affect VAX DSM system operation. **Recommended changes from the VMS default values are indicated in boldface.**

- Buffered I/O Byte Count Limit (BYTLM)

  The BYTLM process quota restricts the amount of buffer space that a process can use for various operations, such as writes to terminals or mailboxes. Accessing files, particularly large files, also uses up the BYTLM quota. The default byte count limit is not adequate for VAX DSM operations because of VAX DSM's heavy use of buffer space for writes. Significantly increase the value of this limit. **A value of 40960 is recommended.**

- Enqueue Quota (ENQLM)

  The ENQLM process quota establishes the maximum number of locks that a process can own. **Set this limit to 300 on VAX DSM user accounts. Set this limit to 1000 on the VAX DSM manager account.**

- File and Logical Link Limit (FILLM)

  The FILLM process quota limits the number of files that a user's process can have open at one time. This limit includes the number of network logical links that can be active at the same time.

  VAX DSM processes must have a sufficiently high FILLM quota to open all database files in order to successfully access the VAX DSM global system. **Set this limit to 150 for all VAX DSM users.** This value should be higher if your application heavily uses RMS globals or other non-database files.

- AST Queue Limit (ASTLM)

  The Incremental Backup Utility (^BACKUP) requires that the ASTLM process quota be at least double the direct I/O limit process quota (DIOLM) for the process performing the incremental backup. Use the following formula to determine the ASTLM value needed for the incremental backup process:

  ```
  ASTLM  ≥  DIOLM * 2
  ```

- Default Working Set Size (WSDEFAULT)

  The WSDEFAULT process quota sets the initial working set size for a user's process. The appropriate size depends on the number of buffers used by the application.

  Use the ^MEMREQ Utility to determine the minimum default working set size (WSDEFAULT) value for processes using nonpaged sections.

- Working Set Size Limit (WSQUOTA)

  The WSQUOTA process quota restricts the size to which the working set of a user's process can expand when systemwide memory demand is significant. The proper value for this limit is highly sensitive to the characteristics of the configuration you are using. The number of disk buffers is the largest contributor to working set requirements. Symbol table and source buffer sizes also contribute.

  If you use nonpaged system memory for your disk cache and mapped routine sections, you can significantly reduce your working set size requirements in environments that require a large number of disk buffers.

  Use the ^MEMREQ Utility to determine the minimum working set limit (WSQUOTA) for processes. Note the amount of paging that occurs while running your application, and increase this value if paging is excessive.

- Working Set Extent Limit (WSEXTENT)

  The WSEXTENT process quota specifies the maximum size to which a user's physical memory usage can grow when the system has excess free pages. WSEXTENT is always greater than or equal to WSQUOTA.

  Use the ^MEMREQ Utility to determine the minimum working set extent limit (WSEXTENT) value for processes using nonpaged sections.

For more information on using DSM to evaluate process quota requirements, see the section ''Using the ^MEMORY Utility'' in this chapter. For more information about process quota limits, see the *Introduction to VMS System Management*.

SYSGEN Parameters for VAX DSM Configurations

VAX DSM requires modified values for a few VMS SYSGEN parameters.

This section lists only those parameters that specifically affect the DSM user environment (either the programmer or the application). For a complete description of each VMS SYSGEN parameter, see the *VMS System Generation Utility Manual*.

The ^MEMREQ Utility displays suggested minimum values for the following parameters based on any configuration that you have previously set up using ^CONFIG.

WSMAX
This parameter should be greater than WSEXTENT (working set extent) for all VAX DSM accounts.

If the WSQUOTA and WSEXTENT values assigned to individual users in the UAF are lower than WSMAX, they are used as maximums.

BALSETCNT
This parameter sets the maximum number of processes you can have resident in memory on your VMS system. If more processes than BALSETCNT log in, the system swaps out other processes to make memory available. Swapping adversely affects performance.

You may need to increase this parameter for VAX DSM if you intend to support a large number of VAX DSM processes.

MAXPROCESSCNT
This parameter sets the maximum number of processes allowed on the system. Make MAXPROCESSCNT at least 1 greater than BALSETCNT.

LOCKIDTBL
This parameter may have to be increased for VAX DSM, particularly if you are using cluster-mounted volume sets. Use the ^MEMREQ Utility to determine the minimum amount to add to this value. To determine the optimum value, observe the number of lock entries under heavy load by running the LOCK option of the VMS Monitor Utility.

LOCKIDTBL_MAX
Whenever you increase LOCKIDTBL, increase LOCKIDTBL_MAX by the same amount. If you are using cluster-mounted volume sets, set LOCKIDTBL_MAX to its maximum value of 65535 entries.

REHASHTBL
Whenever you increase LOCKIDTBL, increase REHASHTBL by the same proportion, and then round the number to the next higher power of two.

GBLSECTIONS
This parameter sets the number of global section descriptors allocated in the system at bootstrap time.

The following describes other requirements for setting GLBSECTIONS:

- If you use nonpaged memory for your DSM global buffers and mapped routine sections, you need one global section for DSM memory management, plus one global section for each mapped routine file.
- If you are using paged memory for these sections, you need one global section for DSM memory management, plus one section for the DSM global buffers, plus two sections for each mapped routine file.
- The ^%JOB Utility requires two global sections.

- One section is required if you enable configuration access control.
- Transaction processing requires an additional global section.

Use the ^MEMREQ Utility to determine the number of global sections that you require when you start up your VAX DSM configuration.

GBLPAGES
This parameter sets the number of global page table entries allocated at bootstrap time.

If you are using nonpaged memory for your DSM global buffers and mapped routine sections, you need to allocate only enough global page table entries to map the DSM memory management and mapped routine data structures. That is, eight plus one for each mapped routine section. If you are using paged memory for your DSM global buffers and mapped routine sections, you need to allocate enough global page table entries to map all of the global buffer section (DSM$CACHE) and each of the mapped routine sections.

Use the ^MEMREQ Utility to determine the number of global page table entries that are required when you start up your VAX DSM configuration.

GBLPAGFIL
This parameter defines the maximum number of systemwide pages allowed for global page-file sections (scratch global sections that use the primary page file for backing storage). If you are using paged memory for your DSM global buffers, you need to increase the current value of this parameter. Use the ^MEMREQ Utility to determine how much to add to this parameter.

If you increase the value of this parameter, you may need to increase the size of the system page file, SYS$SYSTEM:PAGEFILE.SYS, by the same number of blocks. For more information about SYS$SYSTEM:PAGEFILE.SYS, see the *VMS System Generation Utility Manual*.

LRPCOUNT
This parameter establishes the number of large request packets allocated to the LRP lookaside list. VAX DSM DDP allocates and uses LRPs for DDP buffers. If LRPCOUNT is too low, system performance can be adversely affected.[1]

PQL_DWSDEFAULT, PQL_DWSEXTENT, PQL_DWSQUOTA
These parameters define the working set sizes that are used by detached processes that are started using the DCL run command. The ^MEMREQ Utility suggests minimum values for these parameters.

[1] Use the VMS Monitor Utility or the DCL SHOW MEMORY/POOL/FULL command to determine the use of nonpaged dynamic pool, and of small, I/O, and large request packets.

Using the ^MEMORY Utility

Use the Memory and System Parameter Evaluation Utility (^MEMORY) to determine the nominal SYSGEN, process quota, and DSM physical memory values that you need to run a VAX DSM configuration efficiently.

Enter the following to run ^MEMORY:

```
>D ^MEMORY Return
```

VAX DSM displays the following menu:

```
Physical Memory Management Menu

   1.  REQUIREMENT          (^MEMREQ)
   2.  USAGE                (^MEMUSE)

Select Option >
```

The ^MEMREQ Utility examines a specific configuration and determines the amount of memory and other system resources that are required. It also evaluates the amount of memory that is used by each individual process and suggests process quota requirements (WSMAX, WSQUOTA, and WSEXTENT).

Use the ^MEMUSE Utility to determine the physical memory requirements for your configuration. For more information about the ^MEMUSE Utility, see the section ‘‘Configuring Nonpaged Memory.’’

You can run the ^MEMREQ Utility during ^CONFIG when you have completed the definition of a configuration, or by a direct call. In both cases, it is not necessary for the configuration you are evaluating to be running. However, ^MEMREQ requires that the configuration is completely defined. All volume set files, before-image journal files, and routine mapped files specified in the configuration must be present for the ^MEMREQ Utility to work properly.

To invoke ^MEMREQ from the ^CONFIG Utility, answer YES to the show requirements question after the configuration has been built. For example:

```
>D ^CONFIG Return

Would you like extended help [Y OR N] ? <N> Return
Enter the configuration identifier <PAYROLL> Return
.
.
.
Show configuration memory requirements [Y OR N] ?<N>Y Return
```

If you invoke ^MEMREQ directly, you need to specify the configuration for which you want to evaluate SYSGEN and memory requirements.

```
>D ^MEMREQ Return

     DSM Configuration Memory Requirements

Enter the configuration identifier   <PAYROLL> Return
```

The ^MEMREQ Utility displays the parameters on several pages. Each page displays a group of comparable parameters, such as physical memory requirements or minimum process quota values.

The first page displays a summary of SYSGEN parameters and suggested values that are directly related to the efficient operation of the running DSM configuration:

```
              SYSGEN requirements for configuration PAYROLL
    Parameter Name                Value
❶ Minimum SYSGEN Values
    BALSETCNT                       50
    LOCKIDTBL_MAX                65535
    MAXPROCESSCNT                   52
    PQL_DWSDEFAULT                 287
    PQL_DWSEXTENT                  387
    PQL_DWSQUOTA                   337
    VIRTUALPAGECNT                3325
    WSMAX                          387
❷ Configuration Resource Requirements
   ----------------------------------
    GBLPAGES              add      264
    GBLPAGFIL             add      264
    GBLSECTIONS           add        3
    LOCKIDTBL             add      160
    SPTREQ                add      221
    SRPCOUNTV             add      160

❸ Allocated Nonpaged Memory
    DSM/PHYSICAL_PAGES             221

  Press RETURN to continue, '^' to exit
```

The following sections describe SYSGEN requirements:

❶ Minimum SYSGEN Values

You must set the SYSGEN parameters listed in this section to at least the displayed values.

❷ Configuration Resource Requirements

Add at least the values displayed in this section to whatever is required by the VMS operating system for basic system operation (usually these are the current SYSGEN values).

For more information on the VMS System Generation Utility (SYSGEN) and SYSGEN parameters, refer to the *VMS System Generation Utility Manual*.

Note that the values displayed for the SYSGEN parameters are minimum suggested values. Larger values for the parameters displayed in the minimum requirements section are typically used.

Allocated nonpaged memory (❸) refers to the amount of nonpaged memory that you must allocate, using the DSM/PHYSICAL_PAGES=nnnn command line, before you can start this configuration. For more information on how to set up nonpaged memory for use with VAX DSM, see the section ‘‘Configuring Nonpaged Memory’’ in this chapter.

The next page displayed by the utility contains minimum suggested process quota requirements.

```
                Process Quota requirements for configuration PAYROLL
   Process Quota                  Value
   BYTLM                          40960
   ENQLM                            300
   FILLM                            150
   WSDEF                            500
   WSEXT                           1200 **
   WSQUO                           1000

** NOTE:  This is a suggested minimum value for WSEXTENT. If you
          notice excessive paging while the configuration is in use,
          use a higher value for WSEXTENT. You may also want to
          consider the use of Nonpaged Memory for global buffers and
          mapped routine sections.
```

The VAX DSM environment manager can give all users of the configuration a minimum of these quota values. Using smaller values than those displayed affects performance of individual users and can preclude the user's ability to use VAX DSM at all. Larger values can be used, usually at the expense of some amount of system memory. For more information on setting up process quotas for users, see the *VMS Authorize Utility Manual*.

Note that, if the WSEXTENT value becomes large, ^MEMREQ suggests that you use nonpaged physical memory for the global buffers. Using nonpaged memory for global buffers and mapped routine sections enables you to reduce working set sizes for individual users while improving system performance.

The summary reports for setting process quotas and SYSGEN parameters are now complete. This constitutes the majority of information that you need to adjust system parameters. If you would like to see a breakdown of how memory is used by VAX DSM, answer YES to the next question displayed.

The first page of the detailed report is a summary of the resource requirements for the shared sections (the disk buffers, mapped routine sections, and other shared data). The SPTREQ, GBLPAGES, GBLPAGFIL, and GBLSECT values are the amounts that were used to compute summary values previously reported. The value indicated by the /PHYSICAL_PAGES=nnnn command line qualifier indicates the amount of nonpaged memory that should be set aside for the exclusive use of this configuration.

The following is an example of the next ^MEMREQ page:

```
Display a detailed report of memory utilization ? [Y or N] <N>Y Return

              Shared memory parameters for configuration PAYROLL
Section Name   Pages  DSM/PHYS=x  SPTREQ  GBLPAGES  GBLPAGFIL GBLSECT
DSM$CACHE        731       ***     ***       739        739       2
DSM$BAGAAA_SEC   123       123     123         1          1       1
%JOB             256       ***     ***       256        256       2

TOTAL           1110       123     123       996        996       5

** NOTE: This configuration requires 123 pages of real physical
         memory. If you have not done so already, please allocate
         at least this many pages using the DCL command line:

         $ DSM/PHYSICAL_PAGES=123

         You need the CMKRNL and SYSGBL privileges.

Press RETURN to continue, '^' to exit
```

The next display is a summary of the virtual memory requirements for the configuration. The values displayed here were used to calculate the suggested SYSGEN parameters on the first display. Therefore, the configuration NEW, defined for 32 users, might display the following:

```
                Virtual memory requirements for configuration NEW
     Section Name              Pages    x 32 users
❶ Shared sections
     DSM Shared Image (Code)     886           886
     Read-only VMS pages        1385          1385
     PFN Mapped Gbl Sections     123           123
     Read/Write Gbl Sections     996           996
❷ Private sections
     DSM Shared Image (Data)      73          2336
     Process-writable VMS pages  255          8160
     Symbol Table                200          6400
     Routine Buffers              41          1312

     Total                      3703         21342

  Press RETURN to continue, '^' to exit
```

The display is divided into two sets of values:

❶ Shared sections

Shared sections are pieces of virtual memory that different processes can access in common.

❷ Private sections

Private sections are those portions of virtual memory that are owned and used separately by each individual process.

In the previous example, the shared memory requirement for the DSM image is 886 pages. Because this is a shared section, all 32 users share the same copy of the image, and only 886 pages of memory are required for all of them. Each process, however, has its own (unshared) symbol table. Because the symbol tables are not shared, each process needs its own memory for the symbol table and has 32 users * 200 pages or 6400 pages total.

Note that ^MEMREQ assumes that DSM has been installed as a shared image, using the VMS Install Utility.

Modifying VMS System Parameters

Once you have collected the necessary SYSGEN parameter value adjustments using the ^MEMREQ Utility, use the VMS Autogen Utility to modify the relevant SYSGEN parameters.

- If you plan to have only one configuration running on your system, run the ^MEMREQ Utility for that configuration.

  For all parameters listed under the heading ''Minimum SYSGEN Values,'' use the reported value only if it is higher than the current SYSGEN value.

  For all parameters listed under the heading ''Configuration Resource Requirements,'' use SYSGEN to determine if there are sufficient system resources to support the configuration. If additional resources are necessary, sum the required value with the current SYSGEN value to produce the new SYSGEN value.

- If you plan to have multiple VAX DSM environments running VAX DSM configurations at the same time, you need to run ^MEMREQ on each configuration.

  For all parameters listed under the heading ''Minimum SYSGEN Values,'' select the highest value reported from all configurations as the new SYSGEN value, only if it is higher than the current SYSGEN value.

  For all parameters listed under the heading ''Configuration Resource Requirements,'' use SYSGEN to determine if there are sufficient system resources to support the configuration. If additional resources are necessary, sum the values for each parameter for all configurations, and add this sum to the current SYSGEN value.

If you need to inspect the current value for any SYSGEN parameter, use SYSGEN. For example, to inspect the value of the SPTREQ parameter, use the following command sequence:

```
$ RUN SYS$SYSTEM:SYSGEN Return
SYSGEN> SHOW SPTREQ Return
```

Edit the file called SYS$SYSTEM:MODPARAMS.DAT to record site-specific SYSGEN parameter values. AUTOGEN uses this file to establish system parameters. For more information, see the *Introduction to VMS System Management*.

The following is an example of how some SYSGEN values are specified in MODPARAMS.DAT:

```
!
! Site specific AUTOGEN data file. In a VAXcluster where a common
! system disk is being used, this file should reside in
! SYS$SPECIFIC:[SYSEXE], not a common system directory.
!
! Add modifications that you wish to make to AUTOGEN's hardware
! configuration data, system parameter calculations, and page,
! swap, and dump file sizes to the bottom of this file.
!
! Modification history:
!
! 8-September-1989  RSC - Digital
! Recommendations from ^MEMREQ
!
GBLPAGES = 10000
GBLPAGFIL = 9000
GBLSECTIONS = 200
WSMAX = 2048
LOCKIDTBL = 4300
MAXPROCESSCNT = 402
BALSETCNT = 400
VIRTUALPAGECNT = 30000
SPTREQ = 16000
```

Use AUTOGEN to incorporate these values and then reboot the system with the new values for these parameters. The VMS Autogen Utility uses the values supplied within MODPARAMS.DAT to adjust other related parameters in a complementary fashion. Consequently, avoid using SYSGEN to directly manipulate SYSGEN parameters.

Run AUTOGEN using the following command line to modify the system parameters and reboot the system:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT Return
```

Page and Swap File Size

Sufficient space in the VMS page and swap files is critical to system performance. At the extreme, insufficient swap file space can cause processes to enter the RWSWP state and prevent other DSM users from accessing DSM globals. Insufficient page or swap file space can cause the system to hang.

Pageable global pages allocated for the Write Demon buffers are paged against the primary page file. This is important to be aware of when you set up secondary page and swap files. It can present a problem if you create a very small primary page file and intend to use a secondary page file for user processes. VAX DSM startup fails if there are insufficient free global pages in the primary page file.

By using the DCL SHOW MEMORY/FILE/FULL command, you can monitor page and swap file usage on a running system. Use the command file SWAPFILES.COM in SYS$UPDATE to modify the size of the primary page and swap files. Adjust the page file so that its size is greater than twice the usage requirements. Adjust the swap file size so that at least one-fourth of the file is unused and reboot the system. For more information, see the *Introduction to VMS System Management*.

If you need to create secondary page or swap files, use the VMS SYSGEN Utility. For the procedures used to set up secondary page and swap files, see the *Guide to Setting Up a VMS System*.

Configuring VAX DSM Shared Memory

When you define a configuration using the ^CONFIG Utility, you can elect to use *nonpaged memory* to store all data structures that are shared by VAX DSM users. Shared data includes:

- Database disk buffers
- Routine mapped sections
- Volume set and UCI tables
- Miscellaneous process-specific data structures

Nonpaged memory is an alternative to using VMS shared global sections for storage of shared data. Each storage technique has different characteristics which you should consider when setting up your configuration.

- *Shared global sections*

  Shared global sections are segments of VMS virtual memory that are established at VAX DSM startup time.

  — Shared global sections use memory in a dynamic fashion. Virtual memory is allocated at VAX DSM startup and deallocated at shutdown.

  — Shared global section memory pages are pageable. Memory utilization for VAX DSM use is traded against usage of memory by other activities on the system.

  — The working set size of VAX DSM processes is highly sensitive to the size of the shared global section. Large memory sections increase the amount of paging incurred by VAX DSM users if their working set sizes are too small. Large working set sizes increase real memory usage by each VAX DSM user and require larger page and swap file allocations.

— Use of shared global sections may require modifications to the VMS SYSGEN parameters, GBLPAGES, GBLPAGFIL, and GBLSECTIONS, depending on the size of disk cache and number of mapped routine sets.

— The Write Demon is responsible for creating and initializing the shared global section during configuration startup.

- *Nonpaged memory*

  Nonpaged memory uses dedicated physical memory for shared data and mapped routine sets.

  — Nonpaged memory is preallocated and dedicated for use by VAX DSM. Once it is allocated, it cannot be deallocated or used for anything but VAX DSM shared data. Memory is allocated directly from the VMS free list.

  — Nonpaged memory is not paged, regardless of the number of buffers configured. Overall process paging is significantly reduced.

  — The working set size of VAX DSM processes is not increased significantly if you increase the size of the shared data or buffers. This decreases real memory utilization by each VAX DSM process, and reduces the size of the page and swap files.

  — You must modify the VMS SYSGEN parameter SPTREQ and allocate physical memory during VMS system startup.

  — The Write Demon is responsible for partitioning a segment of the preallocated nonpaged memory for use with the starting configuration.

It is recommended that you use shared global sections in situations where flexibility and balanced resource sharing are the major objectives. Support for shared global sections is intended for development or test environments where configuration characteristics are rapidly changing. Also consider use of global sections if you primarily use your VMS system for anything other than a VAX DSM application.

Use of nonpaged memory is recommended in situations where performance and efficient use of memory resources is necessary to support a large capacity VAX DSM application. Support for nonpaged memory is primarily intended for use in stable, production-level application environments where memory requirements, work load, and capacities are well known. If you primarily use your VMS system to support a VAX DSM application only, consider using nonpaged memory for shared data.

Configuring Nonpaged Memory

Options in the ^CONFIG Utility allow you to select which shared sections (disk buffers, mapped routines, or both) will be resident in nonpaged memory. You can select the disk buffers to be resident in nonpaged memory independently of whether any of the mapped routine sections are resident in nonpaged memory.

To use nonpaged memory, you must permanently allocate a fixed portion of contiguous physical memory from the VMS free list for exclusive use by VAX DSM. Allocate this memory with the DSM /PHYSICAL_PAGES=nnnn command line qualifier. The allocated memory serves as a pool of memory that is used by all running VAX DSM configurations that have selected the nonpaged memory option. VAX DSM manages this memory so that it is shared among one or more VAX DSM configurations.

The following command line allocates 10,000 pages (5MB) of physical memory:

```
$ DSM/PHYSICAL_PAGES=10000 Return
```

The VMS CMKRNL and SYSGBL privileges are required to enter this command line. If you reenter the DSM/PHYSICAL_PAGES=nnnn command line with a larger number of pages specified, VAX DSM allocates enough additional pages to reach the requested value.

> **Note:** Once physical memory has been allocated with the /PHYSICAL_PAGES=nnnn command line qualifier, it cannot be deallocated. It can be reused and it can be shared among various VAX DSM environments on the same system, but it cannot be accessed by any facility other than VAX DSM. Be sure to leave enough physical memory so that any other VMS layered products can run properly.

One additional system page-table entry must be configured for each page allocated. Consequently, you must increase the VMS SYSGEN SPTREQ parameter by an equal amount, and then reboot your system.

VAX DSM allocates physically contiguous memory from the VMS free list when you use the DSM/PHYSICAL_PAGES=nnnn command line. Because the free list becomes physically fragmented as the VMS system runs, it is usually necessary to allocate the pages immediately following the reboot of your system. This is especially true if you are attempting to allocate a large number of pages.

You can configure your VMS system to automatically allocate the required pages at VMS boot time. The SYS$STARTUP:DSM$INSTALL.COM procedure is supplied as a template for VAX DSM activities required at VMS system startup. You can customize this procedure for your site. Locate the relevant part of the DSM$INSTALL command procedure that allocates pages for nonpaged memory and make the appropriate modifications. This section is shown in the following example.

```
$ !----------------------------------------------------------------------
$ !
$ !                PHYSICAL MEMORY ALLOCATION
$ !
$ ! If you choose to use physically mapped disk buffers, then you must
$ ! allocate contiguous system memory.  Typically, this is done at VMS
$ ! boot time.  The total number of pages to allocate is determined by
$ ! summing the requirements of each configuration you plan to run
$ ! simultaneously.  In the following command line, remove the comment
$ ! character (!), and edit the line so that:
$ !
$ !   nn = Total # of pages to allocate
$ !
$ ! DSM/MANAGER/PHYSICAL=nn
$ !
$ !----------------------------------------------------------------------
$ !
```

Create your configurations to use nonpaged memory using the following procedure:

1. Determine the number of physical pages and SPTREQs to allocate:
   a. Define all the configurations you plan to run simultaneously. If you plan to run only one configuration that uses nonpaged memory, define only one configuration.
   b. Build any mapped routine sets you plan to install into nonpaged memory.
   c. Initialize the before-image journal file if you plan to use Before-Image Journaling.
   d. For each configuration, use the ^MEMREQ Utility to display the number of physical pages and number of SPTREQs required by each configuration.
   e. Sum the requirements for physical pages and SPTREQs for all configurations.
2. Log in to the VMS system manager account or a privileged account.
3. Edit the SYS$STARTUP:DSM$INSTALL.COM procedure to allocate the appropriate number of physical pages at system startup.
4. Determine the current value of SPTREQ. Edit MODPARAMS.DAT and add the value suggested by ^MEMREQ to the current value of SPTREQ.

See the section ‘‘Modifying VMS System Parameters’’ in this chapter to find out how to modify the SYS$SYSTEM:MODPARAMS.DAT file.

5. Run the VMS AUTOGEN procedure with the REBOOT option.

   The system reboots when AUTOGEN is complete. If you have configured the DSM$INSTALL command procedure to run at system boot time, the nonpaged memory should be available for use by VAX DSM configurations.

If memory is allocated properly, a system global section is created with the name DSM$SYSBUF. Use the VMS Install Utility to verify the existence of DSM$SYSBUF as a system permanent section (PRM SYS), using the following command line in a privileged account:

```
$ INSTALL LIST/GLOBAL Return

        System Global Sections

DSM$SYSBUF     (73EAEFBE)         PRM SYS        Pagcnt/Refcnt=0
.
.
.
```

The DSM$SYSBUF section contains the database that manages the allocation of physical pages for use by various VAX DSM configurations. You can use the ^MEMUSE Utility to display the amount of nonpaged memory that has been allocated on a system and how much is currently in use by each running configuration. Use ^MEMUSE and ^MEMREQ to determine if you have set aside enough physical memory to start up your VAX DSM configurations.

Chapter 5
Optimizing VAX DSM

This chapter contains information on optimizing your VAX DSM system (globals, programs, and applications). For information about database concepts and for more detailed information about managing volume sets and globals, see the *VAX DSM Database Operations Guide*.

Optimizing VAX DSM Applications

To optimize the throughput of a VMS system running a VAX DSM application, do the following:

- Separate your system disk from your global database.
- Make sure your VAX DSM volumes are contiguous files.
- Use VAX DSM volume sets rather than RMS ISAM files for global storage.
- Configure enough disk buffers for the global access rate of your applications.
- Tune VMS parameters for special VAX DSM requirements using the VMS AUTOGEN procedure. Be sure to make the working set quota (WSQUOTA) of the VAX DSM manager adequate for the background processes that operate in the DSM run-time environment.
- Whenever possible, use low values for the following DSM command line qualifiers to reduce memory requirements:

  — /SOURCE_BUFFER_SIZE=n

  — /STACK_SIZE=n

  — /SYMBOL_TABLE_SIZE=n

  For more information on these qualifiers, see the *VAX DSM Programmer's Guide*.

- Map frequently used routines into virtual memory sections, as described in the section ‘‘Using Mapped Routines.’’ Specify the average size of your mapped routines (in pages) using the ^CONFIG or ^RMINS Utility.
- Favor certain VAX DSM application programming techniques over others. For more information on programming optimization techniques, see the *VAX DSM Programmer's Guide*.
- Use nonpaged memory for shared memory. See Chapter 4 for more information.
- Configure a disk cache that is large enough to eliminate global buffer Write Waits.

The following sections describe routine mapping, and outline programming techniques that improve the performance of VAX DSM.

Using Mapped Routines

You can map precompiled VAX DSM routines in a VMS virtual memory section using the VAX DSM Mapped Routine Utilities (^RMAP). Once you map a section, users can access its contents (read only) from the VAX DSM image.

Using mapped routines can improve the response time of your application. Mapped routines improve performance because of the following:

- Virtual memory paging replaces loading a routine from disk. This improves response time because paging I/O is the fastest I/O on a VMS system.
- Because the most frequently used VAX DSM routines tend to remain in physical memory, page faults occur from the free page list rather than from the disk page file.

The virtual memory sections used for mapped routines are permanent system global sections. The system global sections are shared by all processes running under a VAX DSM environment.

Mapped Routine Utilities (^RMAP)

The Routine Mapping Utility (^RMAP) provides the following utilities to help you manage mapped routines:

```
>D ^RMAP Return

Routine Mapping Utilities

    1.  BUILD                    (^RMBLD)
    2.  CONTENTS                 (^RMSHO)
    3.  DELETE                   (^RMDEL)
    4.  DIRECTORY                (MAP^%RD)
    5.  DISABLE ROUTINE          (^RMDIS)
    6.  INSTALL                  (^RMINS)
    7.  SHOW INSTALLED           (MAPSET^RMSHO)

Select Option >
```

Table 5–1 describes each of the Routine Mapping utilities.

Table 5–1 Routine Mapping Utilities

Routine Name	Description
^RMBLD	Builds a file containing precompiled VAX DSM routines that can be mapped into a virtual memory section.
^RMSHO	Lists the names of all routines contained in a specified mapped routine file.
^RMDEL	Allows you to delete a routine set interactively. Normally, mapped routine sections are deleted when you run the ^SHUTDWN Utility. However, you can use ^RMDEL to delete a routine set without shutting down your VAX DSM configuration. Note that the VMS operating system marks sections for deletion, but does not actually delete a section until all processes cease mapping to it.
MAP^%RD	Displays a list of the routines that are mapped for your current application and library directories.
^RMDIS	Disables individual routines in a mapped routine set. Use this utility if, after installing a routine set, you decide that you do not want to run the copy of a routine that is installed in the global section. This is useful when a bug is fixed in a routine, and you want your users to run the new version of the routine, which must be loaded from a volume set, rather than the old version of the precompiled routine, which is in the mapped routine set.
^RMINS	Allows you to install a routine set interactively. You can map routine sets that were not specified in ^CONFIG.
MAPSET^RMSHO	Displays a list of currently installed mapped routine sets.

Determining Memory Requirements of Mapped Routine Sections

Before you can install a permanent global section, you may need to modify the GBLPAGES and GBLSECTIONS SYSGEN parameters to accommodate all mapped routine files that are to be installed. GBLSECTIONS must be large enough to accommodate twice the number of simultaneously installed DSM mapped routine sections, as well as all other VMS installed images and sections. GBLPAGES must be large enough to accommodate the additional total size in pages of all DSM mapped routine sections (plus one extra page per routine section) that are to be installed simultaneously.

To successfully install a paged mapped routine section, you need to be able to contiguously map the section in the VMS global page table. You can use the F$GETSYI lexical function to determine the number of contiguous global pages needed for mapping routines as follows:

1. Find the maximum size of a global section that can be created by entering the following:

   ```
   $ WRITE SYS$COMMAND F$GETSYI("CONTIG_GBLPAGES") Return
   ```

2. Find the allocation size of the mapped routine that you want to install by entering the following:

   ```
   $ DIR/SIZE=ALLOC filename Return
   ```

   where:

   filename is the name of the file that you intend to map as the routine section

3. For the mapped routine section to be installed, the number of contiguous global pages (from step 1) must be at least two greater than the allocation size of the file (from step 2).

4. Also, you must have at least two free global section descriptors available. You can determine this by entering the following:

   ```
   $ WRITE SYS$COMMAND F$GETSYI("FREE_GBLSECTS") Return
   ```

To install a nonpaged mapped routine section (one that uses memory allocated by the /PHYSICAL_PAGES=nnnn command line qualifier) there must be sufficient free pages in the DSM physical memory section. You can determine how much space remains in this section using the ^MEMUSE Utility. Also, you need one free global section descriptor (see step 4).

Use the ^MEMREQ Utility to determine the proper SYSGEN parameters to ensure that you have sufficient memory resources (either paged or nonpaged). First, create all of the mapped routine files that you require and include them in a configuration. Then, run the ^MEMREQ Utility to get the total memory requirements for that configuration. See Chapter 4 for more information on tuning VMS parameters.

Creating and Mapping Routine Sets

Routines are mapped in *sets*. To create a set of routines and map them, do the following:

1. Start up a VAX DSM configuration. Mapped routine sets can only be created when a configuration is running. Note that, you may want to disable user logins from VAX DSM until after you have installed mapped routines.
2. Run VAX DSM with the /MANAGER command line qualifier.
3. Build a mapped routine file with the ^RMBLD Utility. This utility builds a file containing precompiled VAX DSM routines that can be mapped into a virtual memory section.

   The ^RMBLD Utility asks for the UCI and volume set (if more than one volume set is mounted in your configuration) where the routines you want to map reside. The ^RMBLD Utility then calls the ^%RSEL Utility to allow you to select the routines that you want to include in the mapped routine set.

   You can build a mapped routine file for each UCI on each volume set. You can also create mapped routine sets for Library Utilities (those with names starting with a %). A library mapped routine set is associated with a library UCI and volume set.

   Mapped routine files use the following naming convention:

   UCIVOL.VIR

   where:

   UCI is a 3-character UCI name

   VOL is a 3-character volume set name

   VIR is the file extension

   In the following example, ^RMBLD creates a mapped routine set for an application routine directory.

```
>D ^RMBLD Return

Build a DSM Mapped Routine Set

Enter the UCI name in which the routines reside  >PHA Return

routine(s) ? >BILL Return
routine(s) ? >DICT Return
routine(s) ? >MENU Return
routine(s) ? >PAY Return
routine(s) ? >REC Return
routine(s) ? > Return

Using mapped routine file: PHAEMP.VIR

Device and directory to hold this file <DISK$USER:[DSMMGR]> Return
```

```
Writing mapped routines to: DISK$USER:[DSMMGR]PHAEMP.VIR;2

Setting up routine information ...

Writing name subsection
Writing descriptor subsection
Writing program body section

BILL      4248  bytes
DICT      1325  bytes
MENU      3737  bytes
PAY      11417  bytes
REC       8993  bytes

Completing header information

This mapped routine set contains 5 routines.
The average routine size is 5944 bytes.
The average routine will occupy 12 pages of memory.
Installing this mapped routine set requires 65 pages of memory.
```

Note that the ^RMBLD Utility calls the ^%RSEL Utility, which allows you to specify existing mapped routine file names at the routine(s) ? > prompt. This allows you to quickly select all routines that have previously been included in a mapped routine set. To include an existing mapped routine file, enter a left angle bracket (<) before the file specification. For example:

```
>D ^RMBLD Return

Build a DSM Mapped Routine Set

Enter the UCI name in which the routines reside  >PHA Return

routine(s) ? ><DISK$USER:[DSMMGR]PHAEMP.VIR Return
routine(s) ? >^L Return

BILL      DICT      MENU      PAY       REC
routine(s) ? > Return
```

You can use the SAVE^RTHISTP Utility to create a file that contains the names of the most frequently accessed routines. This file name can then be entered at the routine(s) ? > prompt. For example:

```
>D ^RMBLD Return

Build a DSM Mapped Routine Set

Enter the UCI name in which the routines reside  >PHA Return

routine(s) ? ><DISK$USER:[DSMMGR]DSM$RTHIST_PHAEMP.DAT Return
routine(s) ? >^L Return

BILL      DICT      HIRE      MENU      PAY       REC
.
.
.
VACAT

routine(s) ? > Return
```

For more information on the SAVE^RTHISTP Utility, see the section ''Using the Performance Statistics Utility'' in this chapter.

> **Note:** To conserve system resources and memory, map only your most frequently called routines. Mapping routines that are rarely called wastes memory and has little impact on the performance of your system.

4. To include support for mapped routine sets on configuration startup, run ^CONFIG and answer all of the questions pertaining to mapped routines. For example:

```
>D ^CONFIG Return

Build VAX DSM Configuration

      Type ? for HELP at any time


Would you like extended help [Y OR N] ?<N>Y Return
Enter the configuration identifier <PAYROLL> Return
.
.
.
Include support for mapped routines [Y OR N] <N>Y Return

How many routines do you want to install  >1 Return

Enter the number of routine sets you want to have installed
during the startup process. The routine sets should already
be built using the ^RMBLD Utility.


Enter the UCI name in which the routines reside ?  >PHA Return

Enter the UCI name that holds the routines that you want to
include in the mapped section. UCI names must be 3 uppercase
alphabetic characters.

The UCI and volume set names will be used to construct
the VMS global section name given to this routine set.


Enter the volume set name in which the routines reside ?  >EMP Return

Enter the volume set name that holds the routines you want to
include in the mapped section. Volume set names must be 3-uppercase
alphabetic characters.

The UCI and volume set names will be used to construct
the VMS global section name given to the routine set.


Using mapped routine file:  PHAEMP.VIR

Enter the device and directory that contains PHAEMP.VIR
    <DISK$USER:[DSMMGR]> Return

Enter a valid VMS device and directory specification.
```

```
Use nonpaged memory for this routine section [Y OR N] ? <N>Return

Mapped routine sections, like disk cache, may be resident in nonpaged
memory. Because mapped routine sections are shared among many VAX DSM
users, and they can require a significant amount of memory,
using nonpaged memory reduces the amount of page fault activity
each VAX DSM process incurs.

In order to use nonpaged memory, you must have set aside enough
real physical memory to contain the mapped routine sections for this
configuration using the /PHYSICAL_PAGES=nnnn command line qualifier.
Use the ^MEMREQ Utility to determine the nonpaged physical memory
requirements for this configuration.

Answer YES if you have allocated, or will allocate, this memory and
want to use it for nonpaged access to this mapped routine section.
Answer NO if the routine section should be mapped using paged memory.


Page fault cluster size   <8>Return

The page fault cluster size is the number of 512-byte memory
pages that will be paged in when access to a routine within this
routine section causes a page fault. The optimal cluster size
is roughly equivalent to the average size of an application
routine. Assuming that an average routine size is 4KB,
then a good page fault cluster size would be 8.
.
.
.
```

5. Run the ^STU Utility to start DSM and install the mapped sections.

If you do not include support for mapped routines when you ran the ^CONFIG Utility (step 4), you can use the ^RMINS Utility to install the mapped routine set. For example:

```
>D ^RMINSReturn

Install Mapped Routine Sets

Enter the UCI name in which the routines reside ? >PHAReturn

Using mapped routine file: PHAEMP.VIR

Enter the device and directory that contains PHAEMP.VIR <DISK$USER:[DSMMGR]>Return

Use nonpaged memory for this routine section [Y OR N] ? <N>Return
Page fault cluster size   <8>Return

DISK$USER:[DSMMGR]PHAEMP.VIR;2 installed as DSM$00001_MAPR_PHAEMP (62 blocks)
```

Executing Mapped Routines

VAX DSM associates your application routine directory and library routine directory with specific system global sections. The global section names are as follows:

- DSM$nnnnn_MAPR_UCIVOL — mapped routine set section
- DSM$nnnnn_MAPT_UCIVOL — routine disable section

where:

nnnnn	is the unique VAX DSM environment identifier
UCI	is a 3-character UCI name
VOL	is a 3-character volume set name

If a global section is installed for your application routine directory, VAX DSM searches for a routine in the global section. If the routine is not present in the global section, VAX DSM loads the routine from your application routine directory.

If a global section is installed for your library routine directory, VAX DSM searches for library routines (%) in the global section. If the library routine is not present in the global section, VAX DSM loads the routine from your library routine directory.

If you change routine directories by executing the %SETUCI $ZCALL, and there is a mapped routine set installed for your new routine directory, VAX DSM automatically maps to the global section associated with your new routine directory. For example:

```
>D MAP^%RD Return

Output Device ?  <0> Return

No library routines mapped

            VAX DSM mapped application routines   26-JAN-1990 12:08
                       [PHA,EMP]   DSM$00005_MAPR_PHAEMP

BILL      DICT      MENU      PAY       REC

    5 routines
```

The following example displays the mapped routines associated with your new application routine directory.

```
>W $ZC(%SETUCI,"DBS","BJL","R") Return
1
>D MAP^%RD Return

Output Device ?  <0> Return

No library routines mapped

            VAX DSM mapped application routines   26-JAN-1990 12:09
                       [DBS,BJL]   DSM$00005_MAPR_DBSBJL

ADBOOK    STATE     UPCASE

    3 routines
```

Use the MAPSET^RMSHO Utility to display a list of the currently installed routines. For example:

```
>D MAPSET^RMSHO Return

The following mapped routine sets are currently installed:

  Routine Directory      Global Section Name

     [MGR,BJL]           DSM$00005_MAPR_MGRBJL
     [DBS,BJL]           DSM$00005_MAPR_DBSBJL
     [PHA,EMP]           DSM$00005_MAPR_PHAEMP
```

The following example runs ^RMDEL to delete a mapped routine section:

```
>D ^RMDEL Return

Delete a Mapped Routine Section

Enter the UCI name in which the routines reside >MGR Return

DSM$00005_MAPR_MGRBJL         - marked for deletion
```

Disabling Individual Mapped Routines

If you make a change to a routine that resides in an installed mapped routine set, you can disable access to the mapped routine. When a mapped routine is disabled, the precompiled code is loaded from the volume set, rather than from the virtual memory section.

Use the ^RMDIS Utility to disable a mapped routine. For example:

```
>D ^RMDIS Return

Enable/Disable Mapped Routines

Enter the UCI name in which the routines reside  >DBS Return

Enter ^D to display routines mapped to [DBS,BJL]

Enter the name of the routine that you want to enable
or disable mapped access to. The routine must be in the VMS
global section mapped by the UCI and volume set you selected.

Enter ^D to see a display of all routines in this section.
Enter * to select all routines in this section.
Enter ^L to list currently selected routines.

Routine(s)   >^D Return

            VAX DSM mapped application routines   11-MAY-1990 11:09
                       [DBS,BJL]   DSM$00005_MAPR_DBSBJL

ADBOOK    STATE     UPCASE

    3 routines

Routine(s)   >ADBOOK Return
Routine(s)   > Return

Disable or Enable mapped access [D or E]   >D Return

Enter D to disable mapped access to the routine(s), and
force the routine(s) to be loaded from the routine directory.

Enter E to enable mapped access to the routine(s).

ADBOOK      Disabled
```

Do the following to include a new version of a routine in a mapped routine set:

1. Run the ^RMBLD Utility to build a new mapped routine file.
2. Run the ^RMDEL Utility to delete the old mapped routine section.
3. Run the ^RMINS Utility to install the new mapped routine set.

Note that the VMS operating system only marks the old mapped routine section for deletion, but does not actually delete a section until all processes cease mapping to it. Alternatively, you can wait until the next time you shut down and start up your configuration to install the new mapped routine set.

Considerations when Using Mapped Routines

Because mapped routine files contain precompiled code only, VAX DSM must be able to load the source version of a routine to perform the following operations:

- Executing the MUMPS $TEXT function, unless the line starts with a double semicolon (;;). If a line starts with a double semicolon, the VAX DSM precompiler includes the source line in the precompiled routine.
- Generating a VAX DSM error message that includes the line containing the error.

When VAX DSM performs either of these operations, it loads the source code, and then continues to execute code from the mapped section. If VAX DSM performs these operations frequently, it affects the performance of a mapped application.

Global Statistics (^GLSSTA)

Use the Global Statistics Utility (^GLSSTA) to record information on global system performance. Enter the following to run ^GLSSTA:

```
>D ^GLSSTA Return
```

The utility displays the following menu:

```
Global Statistics Menu

    1.  MONITOR                 (MONITOR^GLSSTA)
    2.  RECORD                  (RECORD^GLSSTA)
    3.  REPLAY                  (REPLAY^GLSSTA)

Select Option >
```

The MONITOR^GLSSTA Utility is used to interactively monitor global system performance. The RECORD^GLSSTA Utility writes the global statistics to an output device. The REPLAY^GLSSTA Utility allows you to replay the information that was recorded using RECORD^GLSSTA. You can display Read/Write statistics, Global Access statistics, and Synchronization statistics in separate reports by entering the number of the option you want after the Statistics choice prompt. The default (<*>) displays a report that contains all three sets of statistics.

The reports are displayed interactively on your terminal. To stop the display, press Return. To display another report, enter the number of the option you want after the Statistics choice prompt. To exit the routine, type a circumflex (^) and press Return.

The following is an example of the MONITOR^GLSSTA Utility:

```
>D MONITOR^GLSSTA Return

Monitor Statistics:

Interval: <3> Return

1. Read/Write Statistics
2. Global Access Statistics
3. Synchronization Statistics

Statistics choice: <*> Return
```

After you press Return, the report is displayed on your terminal as shown in Example 5–1.

Example 5–1 ^GLSSTA Report

```
VAX DSM V6.0 Volume Set Statistics    12-AUG-1989 18:08:19.63  0.01
                            CUR     AVE     MIN     MAX   ABS  LAST
❶ Physical Reads           0.00    0.00    0.00    0.00   383     0
❷ Logical Reads         1662.00 1510.91 1662.00 1662.00 54148  4986
❸ Physical Writes         45.67   41.52   45.67   45.67  1873   137
❹ Logical Writes         359.33  326.67  359.33  359.33 10956  1078
❺ Search Last Tries      590.33  536.67  590.33  590.33 25514  1771
❻ Search Last Successes  463.33  421.21  463.33  463.33 18645  1390
❼ Global Accesses        678.67  616.97  678.67  678.67 26335  2036
❽ Global Gets            339.67  308.79  339.67  339.67 13167  1019
❾ Global $DATAs          196.33  178.48  196.33  196.33  4603   589
❿ Global $NEXTs            0.00    0.00    0.00    0.00     0     0
⓫ Global Sets             98.67   89.70   98.67   98.67  8160   296
⓬ Global Kills            44.00   40.00   44.00   44.00   405   132
⓭ Global Seizes           34.67   31.52   34.67   34.67   388   104
⓮ Global Seize Waits       8.67    7.88    8.67    8.67   124    26
⓯ Global Access Waits     21.00   19.09   21.00   21.00   285    63
⓰ Directory Searches      23.00  111.82  123.00  123.00  1209   369
⓱ Block Allocates          9.33    8.48    9.33    9.33   133    28
⓲ Block Deallocates       15.00   13.64   15.00   15.00   141    45
⓳ Global Synch Waits       0.00    0.00    0.00    0.00    29     0
⓴ Global List Waits        9.00    8.18    9.00    9.00   316    27
㉑ Global Buffer Waits     10.67    9.70   10.67   10.67   351    32
㉒ Buffer Write Waits       0.00    0.00    0.00    0.00     0     0
```

The CUR, AVE, MIN, and MAX columns represent per-second averages. The entries displayed are interpreted as follows:

❶ Physical Reads

The number of times a real physical disk read was performed to read a database block into DSM global buffer cache.

❷ Logical Reads

The number of times a database block was required for a global access.

❸ Physical Writes

The number of DSM blocks written to disk by the DSM Write Demon.

❹ Logical Writes

The number of times a disk block was modified by a DSM process and marked as needs writing.

❺ Search Last Tries

The number of times that a DSM process attempted to resolve a global reference by checking the last DSM block that it referenced.

❻ Search Last Successes

The number of times that a DSM process actually resolved a global reference by checking only the last DSM block that it referenced.

❼ Global Accesses

The total number of global GET, SET, KILL, $DATA, and $NEXT operations performed by the DSM global system.

❽ Global Gets

The total number of global variable reads done. For example:

```
S A=^X(4)
```

❾ Global $DATAs

The total number of global variable $DATAs done. For example:

```
S A=$D(^X(4))
```

❿ Global $NEXTs

The total number of global variable $NEXTs done. For example:

```
S A=$N(^X(-1))
S A=$O(^X(""))
```

⓫ Global Sets

The total number of global variable SETs done. For example:

```
S ^X(4)="PATIENT"
```

⓬ Global Kills

The total number of global variable KILLs done. For example:

```
K ^X(4)
```

⓭ Global Seizes

The number of times that a volume set was exclusively owned by a DSM process while it performed a multiple-block update to the DSM database (either a block split or merge operation).

⓮ Global Seize Waits

The number of times that a DSM process had to gain exclusive access to a volume set because other processes were already accessing it concurrently.

⓯ Global Access Waits

The number of times that a DSM process had to wait to access a volume set because another process had it seized (exclusively-owned).

⓰ Directory Searches

The number of times that DSM processes searched the global directory of a volume set in order to satisfy a global reference.

⓱ Block Allocates

The number of DSM blocks added to a DSM global structure, as in the case of a split operation.

⓲ Block Deallocates

The number of DSM blocks removed from a DSM global structure and marked as free, as in the case of a merge operation.

⓳ Global Synch Waits

The number of times that a DSM process had to wait for exclusive access to a data structure other than the actual DSM global buffers.

⓴ Global List Waits

The number of times that a DSM process had to wait before searching for a DSM database block in cache.

㉑ Global Buffer Waits

The number of times that a DSM process had to wait for exclusive access to a cached DSM database block.

㉒ Buffer Write Waits

The number of times that a DSM process had to wait to read a new block into cache because the cache was full of unwritten modified blocks.

High LOGICAL/PHYSICAL READ and LOGICAL/PHYSICAL WRITE ratios indicate that cache efficiency is correspondingly high. Low LOGICAL/PHYSICAL ratios indicate insufficient cache or extremely scattered database references.

A high number of SEARCH LAST SUCCESSES indicates that the application is making good use of the DSM most recently used checking algorithm. Note that if the SEARCH LAST algorithm fails for a global reference, DSM must trace the global reference from the topmost pointer block back to the data level. Because this operation involves upwards of two database block accesses, a SEARCH LAST SUCCESS represents a substantial performance savings.

Any of the Wait statistics (Seize Waits, Access Waits, Synch Waits, List Waits, Buffer Waits, and Write Waits) indicate that processes are contending for various DSM resources. The more that processes are forced to wait, the less concurrence there is in the global system. Furthermore, system overhead is expended when processes are forced to wait to coordinate access to the resources.

Under extreme conditions, buffer Write Waits can cause severe performance degradation. In these cases, eliminating the disk cache usually eliminates these problems.

The global handler arbitrates access on a per-volume set basis. Consequently, you can increase the concurrency and efficiency of the database handler by moving the most frequently modified globals to separate volume sets.

The other statistics are indicators of how much work the global system is doing and are not indicators of database efficiency.

Performance Statistics (^RTH)

The ^RTH Utility provides a graphic display of system usage that can help identify system bottlenecks. Histograms depict global system performance parameters, relative CPU use by routine, relative frequency of routine loads, global references per second by individual global, and DDP activity. You can use this information to decide which routines to map, where to place globals to balance the load, and to identify those routines that are the best candidates for code optimization if system response time or throughput is unacceptable.

Using the Performance Statistics Utility

You can obtain performance statistics on your VAX DSM system by using the Performance Statistics Utility (^RTH). You can access ^RTH from the System Utilities menu (^SYS), or enter the following:

```
>D ^RTH Return
```

The utility displays the following menu:

```
RTH Utilities

    1.  LOG                   (^RTHIST)
    2.  PRINT                 (^RTHISTP)
    3.  SAVE                  (SAVE^RTHISTP)
    4.  STATUS                (STATUS^RTH)
    5.  STOP                  (HALT^RTH)

Enter option >
```

The following conventions apply to the ^RTH Utility options:

- Press Ctrl/z or Ctrl/c to exit from the option.
- Type a question mark (?) and press Return for help at any input prompt.
- Type a circumflex (^) and press Return to repeat the previous input prompt.
- Type a circumflex and the letter L (^L) and then press Return to list available choices where appropriate.

The LOG option (^RTHIST) allows you to log performance data for a certain period of time (a session). The ^RTHIST option creates five different reports, which are described in the following sections.

To run ^RTHIST, enter the following:

```
>D ^RTHIST Return

Log System performance statistics

How many logging sessions <1> Return

How many minutes will each logging session last <1> Return

Delay how long (minutes) between sessions <5> Return

Enter description field for report headings
           <CASEY 18-JAN-90 1:18:20 PM> Return

Begin collecting after   <18-JAN-1990 13:18:20> Return

The statistics logging session will now be started with a JOBBED process.
The first session will be filed as #1.
Collection will begin at 18-JAN-1990 13:18:20.
You can check on the status of the statistics session by running STATUS^RTH.
Do HALT^RTH to signal the statistics logger to stop after its next session.

                                                    Press RETURN to continue
```

You need to enter the number of logging sessions you want, the number of minutes each logging session should last (up to 480 minutes), the delay (up to 1440 minutes) between sessions, a descriptive text field for the report heading, and a date and time to begin collection.

The PRINT option (^RTHISTP) lets you print out the reports that contain the data that was logged using the LOG option. Note that the reports are 132 columns wide. You can display the reports on your terminal, or output them to a specific device. You can print multiple logging session reports in a single print session. If you choose to print multiple logging sessions, a consolidated report is also produced.

The SAVE option (SAVE^RTHISTP) allows you to save the names of the most frequently loaded routines during the logging session for later use in building mapped routine global memory sections. The SAVE^RTHISTP Utility divides the routine names into a *load distribution*, which lists the number of routines that represents each 10 percent increment of total routine loads. You can then choose what percentage of total routine loads you want to cover with routine mapping.

An example distribution is shown in the following example:

```
>D SAVE^RTHISTP Return

SAVE FREQUENTLY ACCESSED ROUTINES TO FILE(S)

Node Identifier   <CASEY> Return

Which statistics session(s)  <1> Return

3   routines account for  30 percent of loads. ( >=    0.49 loads/second)
15  routines account for  40 percent of loads. ( >=    0.13 loads/second)
30  routines account for  50 percent of loads. ( >=    0.11 loads/second)
70  routines account for  60 percent of loads. ( >=    0.11 loads/second)
90  routines account for  70 percent of loads. ( >=    0.10 loads/second)
120 routines account for  80 percent of loads. ( >=    0.10 loads/second)
200 routines account for  90 percent of loads. ( >=    0.05 loads/second)
320 routines account for 100 percent of loads. ( >=    0.02 loads/second)

Cover what % of total routine loads with routine mapping?  >80 Return

Saving 120 routine names (loaded more than    0.10 times/second).

Output Device  ? <DSM$RTHIST_PHAEMP.DAT> Return

FILE:  DSM$RTHIST_PHAEMP.DAT   - 120 routine names saved.
```

After choosing the sessions to use as a basis for mapping, a routine load distribution is displayed and you are asked to decide the percentage of total routine loads to be covered by mapping. The previous example covers 80% of the routines and saved 120 routine names. The files built for each volume set and UCI can be used as input at the routine selection prompt of the ^RMBLD Utility.

This routine allows you to automate and improve the accuracy of choosing the appropriate routines for inclusion into a mapped routine set. Only the minimal number of routines are mapped to produce the greatest efficiency.

The STATUS^RTH option shows if any statistics logging sessions are currently active.

The HALT^RTH option signals a statistics logging process to stop collecting at the end of the current logging session.

Data gathered on many statistics is displayed as histograms by the ^RTHIST option.

The following is a list of the types of data that ^RTHIST reports on:

- Database event counts
- Derivative ratios
- Routine execution time
- Routine call counts
- Global reference counts
- Distributed Data Processing (DDP) activity

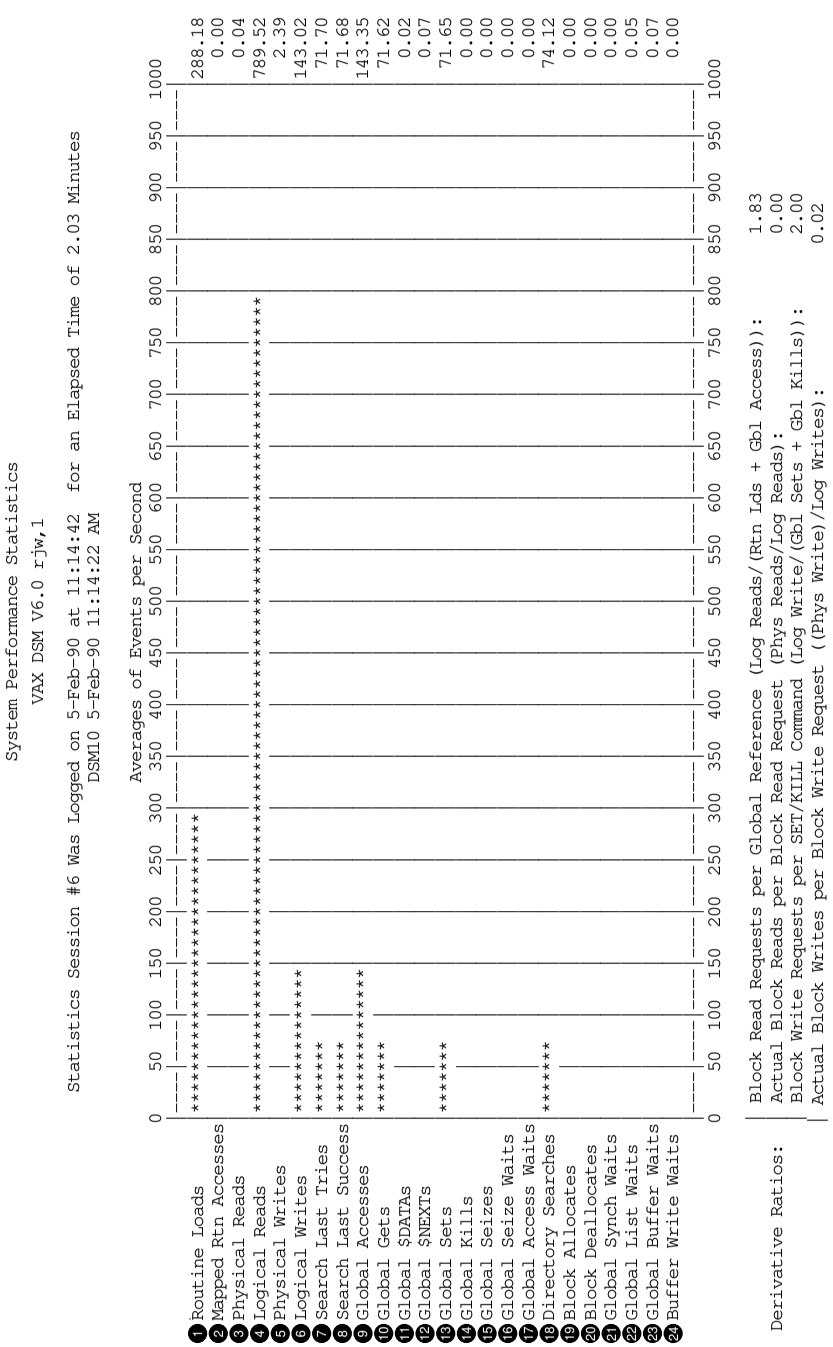
The following sections describe each of the ^RTH histograms.

Database Event Counts and Derivative Ratios Histogram

The Database Event Counts and Derivative Ratios Histogram displays global and routine statistic information that the database access handlers automatically accumulate. The data represents averages of events per second.

Figure 5–1 shows an example of the histogram that displays the database event counts and derivative ratios based on the database event count statistics.

Figure 5–1 Database Event Counts and Derivative Ratios Histogram



The database event count statistics displayed in this histogram are described as follows:

❶ Routine Loads

The number of times that routines are loaded because of DO, GOTO, ZLOAD, and ZSAVE commands.

❷ Mapped Rtn Accesses

The count of accesses to routines that are mapped in memory.

❸ Physical Reads

The number of times a real physical disk read was performed to read a database block into DSM global buffer cache.

❹ Logical Reads

The number of times a database block was required for a global access.

❺ Physical Writes

The number of DSM blocks written to disk by the DSM Write Demon.

❻ Logical Writes

The number of times a disk block was modified by a DSM process and marked as needs writing.

❼ Search Last Tries

The number of times that a DSM process attempted to resolve a global reference by checking the last DSM block that it referenced.

❽ Search Last Successes

The number of times that a DSM process actually resolved a global reference by checking only the last DSM block that it referenced.

❾ Global Accesses

The total number of global GET, SET, KILL, $DATA, and $NEXT operations performed by the DSM global system.

❿ Global Gets

The total number of global variable reads done. For example:

```
S A=^X(4)
```

⓫ Global $DATAs

The total number of global variable $DATAs done. For example:

```
S A=$D(^X(4))
```

⓬ Global $NEXTs

The total number of global variable $NEXTs done. For example:

```
S A=$N(^X(-1))
S A=$O(^X(""))
```

⓭ Global Sets

The total number of global variable SETs done. For example:

```
S ^X(4)="PATIENT"
```

⓮ Global Kills

The total number of global variable KILLs done. For example:

```
K ^X(4)
```

⓯ Global Seizes

The number of times that a volume set was exclusively owned by a DSM process while it performed a multiple-block update to the DSM database (either a block split or merge operation).

⓰ Global Seize Waits

The number of times that a DSM process had to gain exclusive access to a volume set because other processes were already accessing it concurrently.

⓱ Global Access Waits

The number of times that a DSM process had to wait to access a volume set because another process had it seized (exclusively-owned).

⓲ Directory Searches

The number of times that DSM processes searched the global directory of a volume set in order to satisfy a global reference.

⓳ Block Allocates

The number of DSM blocks added to a DSM global structure, as in the case of a split operation.

⓴ Block Deallocates

The number of DSM blocks removed from a DSM global structure and marked as free, as in the case of a merge operation.

❷❶ Global Synch Waits

The number of times that a DSM process had to wait for exclusive access to a data structure other than the actual DSM global buffers.

❷❷ Global List Waits

The number of times that a DSM process had to wait before searching for a DSM database block in cache.

❷❸ Global Buffer Waits

The number of times that a DSM process had to wait for exclusive access to a cached DSM database block.

❷❹ Buffer Write Waits

The number of times that a DSM process had to wait to read a new block into cache because the cache was full of unwritten modified blocks.

Derivative ratios show the effectiveness of the optimization algorithms of both the disk cache and the Global Handler. These ratios are derived from the database event count statistics. Figure 5–1 shows the derivative ratio statistics. The following describes these statistics:

- Block Requests for each Global Reference — shows the effectiveness of the Global Handler. In general, global and routine references require more than one block request to get the data. The Global Handler uses optimization algorithms to minimize the number of these block requests. When successive references (within a job) are to the same global, the second and following references can require fewer block requests. If global references within a job are grouped by global name, the ratio declines to approach the value of 1.
- Actual Block Reads for each Read Request — shows the effectiveness of the disk cache in minimizing disk reads.
- Block Write Requests for each SET command — must be at least 1. This ratio can be expected to increase as the rate of growth of the database increases. When a global block overflows, a split occurs, causing at least four block writes to the following:
  - — Two resulting blocks
  - — Ancestor pointer block
  - — Map block
- Actual Block Writes for each Write Request — shows the effectiveness of the disk cache and the Write Demon in minimizing disk writing.

Individual Routine CPU Time as a Percentage of Total Routine CPU Time Histogram

The Individual Routine CPU Time as a Percentage of Total Routine CPU Time Histogram shows the relative CPU cost of routines. This report can point to routines that are candidates for code optimization if your system is experiencing response time or throughput problems.

Figure 5–2 shows an example of the histogram that displays the time spent executing routines as a percentage of the total CPU time for the logging session.

Figure 5–2 Individual Routine CPU Time as a Percentage of Total Routine CPU Time Histogram

```
                                  System Performance Statistics
                                        VAX DSM V6.0 rjw, 1

                 Statistics Session #6 Was Logged on 5-Feb-90 at 11:14:42  for an Elapsed Time of 2.03 Minutes
                                      DSM10 5-Feb-90 11:14:22 AM

                                 Individual Routine CPU Time as a Percentage of Total Routine CPU Time
UCI       Routine   0    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
---       -------   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
                    |                                                                                                                
RJW,AAA   RH1       |****|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  3.90
          RH2       |********|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  7.98
          RH3       |************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  12.09
          RH4       |****************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  15.92
          RH5       |********************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  19.90
          RHTEST    |****************************************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  39.86
          AAA Pcnt  |******************************************************************************************************* 99.66
                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
RJW,MGR   %SYSROU   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.21
          RTH       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.05
          RTHIST    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.01
          RTHIST1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.06
          MGR Pcnt  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.34
                    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
                    0    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100

                    ---------------------------------------------
                    | Total routine CPU time   1:52             |
                    ---------------------------------------------
```

Individual Routine Loads as Percentage of Total Histogram

The Total Routine Loads Histogram shows which routines have been loaded as the result of a DO, GOTO, or ZLOAD command, or as the result of a trap via the $ZTRAP mechanism. Choose routines that display the highest percentage of loads to maximize the performance improvement of routine mapping.

Figure 5–3 shows an example of the histogram that displays the individual routines as a percentage of total routine loads.

Figure 5–3 Individual Routine Loads as a Percentage of Total Routines Loads Histogram

```
                                  System Performance Statistics
                                       VAX DSM V6.0 rjw, 1

                 Statistics Session #6 Was Logged on 5-Feb-90 at 11:14:42  for an Elapsed Time of 2.03 Minutes
                                    DSM10 5-Feb-90 11:14:22 AM

                                 Individual Routine Loads as a Percentage of Total Routine Loads
UCI       Routine   0    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
---        -------  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
RJW,AAA   RH1      |***|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  3.34
          RH2      |*******|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  6.67
          RH3      |**********|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  9.97
          RH4      |*************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 13.32
          RH5      |****************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 16.69
          RHTEST   |**************************************************|    |    |    |    |    |    |    |    |    |    | 49.99
          AAA Pcnt |*****************************************************************************************************| 99.98
                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
RJW,MGR   %SYSROU  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.00
          RTH      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.00
          RTHIST   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.01
          RTHIST1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.01
          MGR Pcnt |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  0.02
                   |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
                   0    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
```

Global References Per Second Histogram

The Global References Per Second Histogram is an analysis, by UCI and the global name, of the combined count of all references to global variables. This is actually a global-by-global breakdown of the counts recorded previously as Global GETs, $DATAs, or $NEXTs (no sign following the global name), Global SETs (plus sign (+) following the global name), and Global KILLs (minus sign (–) following the global name).

The statistics area has a limited amount of memory. Each routine and global referenced occupies one cell. When available space is filled, subsequent new unique references are logged in the No Room category.

Figure 5–4 shows an example of the histogram that displays the number of global references per second.

Figure 5–4 Global References Per Second Histogram

```
                                  System Performance Statistics
                                        VAX DSM V6.0 RJW, 1

                 Statistics Session #6 Was Logged on 5-Feb-90 at 11:14:42  for an Elapsed Time of 2.03 Minutes
                                      DSM10 5-Feb-90 11:14:22 AM

                                                Global references per second
UCI       Global    0    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
---       ------    |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
RJW,AAA   RH1       *****    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    4.75
          RH1 +     *****    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    4.75
          RH2       **********    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    9.51
          RH2 +     **********    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    9.51
          RH3       **************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   14.21
          RH3 +     **************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   14.21
          RH4       *******************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   18.97
          RH4 +     *******************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   18.97
          RH5       ************************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   23.77
          RH5 +     ************************|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   23.77
          AAA Total |-------------------------------------------------------------------------------------------------------142.43
                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
RJW,MGR   %MENU     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    0.20
          MGR Total |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|    0.20
                    0    5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
```

Distributed Data Processing Activity Histogram

The Distributed Data Processing Activity Histogram provides a report of DDP traffic that occurs during the ^RTHIST logging session. The following information is supplied in this report about each node in the network:

- Requests received per second
- Requests sent per second
- Packets sent during the session
- Messages per packet
- Bytes per packet
- Requests retried during session
- Requests out of SYNCH
- Messages transmitted per second
- Packets transmitted per second
- Bytes transmitted per second

For more information on these fields and the optimization implications of this information, see the *VAX DSM Database Operations Guide*.

Figure 5–5 shows an example of the report that displays the DDP activity on your system during the logging session.

Figure 5–5 Distributed Data Processing Activity Histogram

System Performance Statistics
VAX DSM V6.0 rjw, 1

Statistics Session #6 Was Logged on 5-Feb-90 at 11:14:42 for an Elapsed Time of 2.03 Minutes
DSM10 5-Feb-90 11:14:22 PM

Distributed Data Processing Activity

Node	Requests Rcvd/Sec	Requests Sent/Sec	Packets Sent	Messages Packet	Bytes/ Packet	Requests Retried	Requests Out-of-Sy	Messages Xmt/Sec	Packets Xmt/Sec	Bytes XMT/Sec
AAA	0	0	0	0.00	0.00	0	0	0	0	0
BBC	0	0	0	0.00	0.00	0	0	0	0	0
CCC	0	0	0	0.00	0.00	0	0	0	0	0
DDD	0	0	0	0.00	0.00	0	0	0	0	0
DGC	0	0	1	1.00	16.00	0	0	0	0	0
EEE	0	0	0	0.00	0.00	0	0	0	0	0
EJM	9	9	1499	1.46	114.95	0	0	18	13	1448
FFF	0	0	0	0.00	0.00	0	0	0	0	0
RJW	0	0	0	0.00	0.00	0	0	0	0	0
TIK	0	0	0	0.00	0.00	0	0	0	0	0
TOTAL	9	9	1500	1.46	114.88	0	0	18	13	1448

Optimizing VAX DSM Based on Performance Statistics

An advantage of the ^RTH Utility is that it gathers statistics while an application is running in a normal day-to-day environment, rather than in a test situation. You can use the statistics to accurately gauge where bottlenecks or performance problems are occurring in the system.

When a system reaches a saturation point, the first question usually asked is whether the system is disk-bound or CPU-bound. The answer can be inferred by examining the ^RTH results in conjunction with the VMS Monitor Utility.

If the system is CPU-bound, you may want to optimize VAX DSM coding to improve performance. You can also refer to the ^RTH statistics to see which routines are consuming the most CPU time, and determine where your effort would produce the best results. Another option for CPU-bound systems is to consider upgrading to a faster CPU.

If the system is disk-bound, you can use these statistics to show how often each global is being referenced, how the references are split between reading and writing, and which routines, when mapped, could best minimize disk references.

Chapter 6
VAX DSM System Utilities

This chapter describes the VAX DSM System Utilities that are useful to VAX DSM managers and operators.

Overview of VAX DSM Utilities

The VAX DSM utilities are a collection of routines provided with your VAX DSM system that perform tasks and provide needed information. The utilities fall into two categories:

- System Utilities
- Library Utilities

In general, Library Utilities serve the needs of application programmers, and System Utilities serve the needs of VAX DSM managers. Any VAX DSM user can access Library Utilities, described in detail in the *VAX DSM Programmer's Guide*. The names of these routines begin with a percent (%) sign. System Utilities, described in this chapter and elsewhere in this manual, are only accessible to users who enter the DSM command with the /MANAGER command line qualifier, or log in to the MGR UCI. These utilities reside in the RMS routine directory, SYS$LIBRARY:MANAGER.DSM, when volume sets are not mounted. Otherwise, these utilities reside in the MGR UCI of the first mounted volume set.

VAX DSM provides access to all System and Library Utilities through a menu. The menu displays options on your terminal, and you select the option you need for a particular task.

The utilities also provide online help. If you are unsure of the appropriate response to a prompt, type a question mark (?) for help. VAX DSM responds by listing your options, or by asking you to identify the option about which you need further information.

Online Help

VAX DSM contains an online help system, the ^%HELP Utility, that provides information about VAX DSM commands and other aspects of the VAX DSM language and operating system.

To access the ^%HELP Utility, type a question mark (?), or type **D ^%HELP** after the VAX DSM prompt (>) as follows:

```
>? Return
>D ^%HELP Return
```

VAX DSM displays the following options:

```
Help is available on the following DSM syntax elements:

COMMANDS        DEBUGGER       EXTERNAL FUNCTIONS     FUNCTIONS
OPERATORS       UTILITIES      VARIABLES

DSM help >
```

After the DSM help > prompt, enter the name of one of the seven categories. The utility then displays a list of specific elements (such as commands) that are included in that category. Enter the name of the specific element (such as a command) that you want more information on.

> **Note:** The routine buffer is overwritten by the online help routines. If you use this online help while developing routines, be sure to save any partial edits with the ZSAVE command before invoking the online help.

Utility Conventions

All utilities use the following conventions, unless the utility description specifies an exception to a convention:

- To get information about a utility or utility prompt, type a question mark (?). In response, the system displays help text.
- Each menu displays a list of numbers followed by options. Options can point to lower level menus or to actual utilities. To select an option, enter either its number or its name (enough characters to distinguish it from other names). The menu also displays the utility routine name in parentheses.

- The correct way to exit a menu is to press Return, or type a circumflex (^) and press Return after a prompt. To exit any utility, use the same procedure. If the utility interprets a Return as a default value, use the circumflex to back up or exit.

- After a utility begins execution, do not stop the utility before execution is complete. You run the risk of leaving files open, leaving a VIEW buffer allocated, or leaving variables undefined. Let the utility run to completion, then rerun it correctly.

- When a utility requests an input or output file, it accepts any valid VMS device and file specification. You can refer to your own terminal as 0, but you must specify all other devices by device name or logical name. If your VMS installation spools the line printer, you can specify LP in the device field of the output file specification. When you include the line-printer name in the file specification (for example, LPA0:EXAMPLE.DAT), the file name appears on the header page of your output.

- For some utilities, such as those that manipulate routines or globals, you may want to specify more than one name in response to a utility prompt. You can use the asterisk (*) as a wildcard character. The asterisk by itself indicates ALL. When the asterisk is the last character in an alphanumeric string, the utility interprets that string as meaning all names with that string as their root. For example, to specify ALP, ALPHA, and ALP123, type ALP*.

  You can also do the following:

  — Type ^D in response to a prompt requesting a routine or global name. This response displays the contents of your global or routine directory (to help you make your selection).

  — Type ^L to produce a list of the routines or globals you have already selected.

  — Enter a range of names. For example, A–FZZZ.

  — To exclude a single name or a range of names from an operation, use a minus sign before the routine names. For example, –A* and –B–CZZ. Note the use of the wildcard.

- VAX DSM always returns numbers in a decimal format. You can, however, convert numbers to other bases using the numerical conversion Library Utilities described in the *VAX DSM Programmer's Guide*.

Running the VAX DSM System Utilities

You can access the VAX DSM utilities in the following ways:

- Enter the following to display a menu of the System Utilities:

  ```
  >D ^SYS Return
  ```

  The utility displays the following menu:

  ```
  System Utilities

      1.  CONFIGURATION MANAGEMENT        (^CONMAN)
      2.  DATABASE SET MANAGEMENT         (^DBSMAN)
      3.  DISTRIBUTED DATA PROCESSING     (^DDP)
      4.  MAPPED ROUTINES                 (^RMAP)
      5.  OPTION FILE MANAGEMENT          (^OPTMAN)
      6.  PERFORMANCE UTILITIES
      7.  SYSTEM MANAGEMENT               (^SYSMAN)

  Select Option >
  ```

- Enter the routine name of the VAX DSM utility directly in Programmer Mode:

  ```
  >D ^CONMAN Return
  ```

- Run VAX DSM in Application Mode using the routine name of the utility as the DSM command parameter:

  ```
  $ DSM/MANAGER ^CONMAN Return
  ```

Configuration Management Utilities (^CONMAN)

Use the ^CONMAN Utilities to create and manage configurations. You can access the following Configuration Management Utilities from the ^CONMAN Utility menu:

Utility	Routine Name
Copy Configuration	COPY^CONFIG
Build Configurations	^CONFIG
Delete Configuration	DELETE^CONFIG
Display Configuration	DISPLAY^CONFIG
Modify Existing Configuration	MODIFY^CONFIG

Utility	Routine Name
After-Image Journaling	^CONAIJ
Command Line Qualifiers	^CONCLQ
Database Set	^CONDBS
DDP Options	^CONDDP
Global Characteristics	^CONGLO
Global Translation	^CONTR
Mapped Routine Sets	^CONMAP
System Options	^CONOPT
Transaction Processing	^CONTP

For information about the ^CONFIG Utility, see Chapter 2 in this manual. The following sections describe the other ^CONMAN Utilities.

Displaying Configurations (DISPLAY^CONFIG)

The following example describes the DISPLAY^CONFIG Utility. It displays information about the configuration that you defined to completion with the ^CONFIG Utility. For example, enter the following to display the PAYROLL configuration:

```
$ DSM/MANAGER Return

          VAX DSM V6.0
            DSMMGR
           [MGR,TES]

>D DISPLAY^CONFIG Return
```

The utility displays the following:

```
*** CONFIGURATION DISPLAY

Enter the configuration identifier <PAYROLL>Return

Output Device ?  <0>Return

                        *** CONFIGURATION PAYROLL ***
System options :
--------------
     Configuration access control enabled        Y
     Number of disk buffers                      1200
     Memory for disk buffers                     Paged
     Maximum number of users                     128
     Restrict use of the VIEW command            NO
     Support RTHIST performance histogram        NO

Database sets:
------------
Database set #1 = EMPLOYEE

     VAX DSM database set
     Not Cluster-Mounted

     EMP  DISK$DATA:[DSM$DSMMGR.DATABASE]EMPVOL1.GLS
          No Before-Image Journaling
          No After-Image Journaling

Configuration PAYROLL is the database set
master for the following database sets:

          NONE

DDP parameters:    DDP not included
--------------

Default global characteristics :
------------------------------
     8 Bit Subscripts              Yes
     Journaling                    No
     Collating sequence            Numeric

     Access privileges for non-% globals :
        System: RWD     World: NONE     Group: NONE     UCI: RWD

     Access privileges for % globals :
        System: RWD     World: R        Group: R        UCI: RWD

Mapped routine sets to install:  mapped routines not included
------------------------------

DSM default command line qualifiers:  none defined

TP options:   TP not included
----------

After-Image Journaling:  not enabled
----------------------
```

Copying Configurations (COPY^CONFIG)

The COPY^CONFIG Utility allows you to copy all of the configuration information for an existing configuration to create another configuration definition.

Enter the following to run COPY^CONFIG:

```
>D COPY^CONFIG Return

This operation creates a new configuration by copying the contents
of an already existing configuration.

Name of configuration to copy:
Enter the configuration identifier >PAYROLL Return

Name of configuration to create:
Enter the configuration identifier >BENEFITS Return

Are you sure you want to copy configuration PAYROLL to BENEFITS [Y OR N] ?<N>Y Return

Copying configuration PAYROLL to a new configuration BENEFITS...

Copy complete
```

This utility is useful for creating configurations for nodes in a VAXcluster environment. You can make copies of an existing configuration and then use the MODIFY^CONFIG Utility to make partial changes.

Modifying Configurations (MODIFY^CONFIG)

The MODIFY^CONFIG Utility allows you to modify parts of an existing configuration without having to run the entire ^CONFIG Utility. Enter the following to run MODIFY^CONFIG:

```
>D MODIFY^CONFIG Return

Modify Configuration Utilities

    1.  AFTER-IMAGE JOURNALING            (^CONAIJ)
    2.  COMMAND LINE QUALIFIERS           (^CONCLQ)
    3.  DATABASE SET                      (^CONDBS)
    4.  DDP OPTIONS                       (^CONDDP)
    5.  GLOBAL CHARACTERISTICS            (^CONGLO)
    6.  GLOBAL TRANSLATION                (^CONTR)
    7.  MAPPED ROUTINE SETS               (^CONMAP)
    8.  SYSTEM OPTIONS                    (^CONOPT)
    9.  TRANSACTION PROCESSING            (^CONTP)

Select Option >
```

Each option allows you to modify existing information. For example, if you select the ^CONDDP Utility, the DDP questions from ^CONFIG are displayed. All of the options, except the ^CONTR Utility, display the same information as ^CONFIG. The ^CONTR Utility displays a menu of utilities and is discussed in detail in the *VAX DSM Database Operations Guide*.

Deleting Configurations (DELETE^CONFIG)

The DELETE^CONFIG Utility lets you delete an existing configuration. Enter the following to run the utility:

```
>D DELETE^CONFIG Return

WARNING *** This operation DELETES an entire configuration

Enter the configuration identifier   >MAIN Return

Are you sure you want to delete the MAIN configuration [Y OR N] ? <N>Y Return

Deleting the MAIN configuration... done.
```

Database Set Management Utilities (^DBSMAN)

Use the ^DBSMAN Utilities for After-Image Journaling, Before-Image Journaling, and database and volume set management. You can access the following utilities from the ^DBSMAN Utility menu:

Utility	Routine Name
After-Image Journaling	^JOURNAL
Before-Image Journaling	
Database Set Utilities	
Volume Set Utilities	^VOLUTL

The following sections describe these utilities.

After-Image Journaling Utilities (^JOURNAL)

The ^JOURNAL Utilities control the use of the VAX DSM After-Image Journaling process. To use the options on the ^JOURNAL menu, you must have the same UIC as the VAX DSM Journal Process. You can access the following After-Image Journaling Utilities from the ^JOURNAL Utility menu:

Utility	Routine Name
Dejournal	^DEJRNL

Utility	Routine Name
Dejournal from a DSM-11 Tape	DSM11^DEJRNL
Dejournal from Journal Log	LOG^DEJRNL
Dejournal Selected Files	FILE^DEJRNL

Utility	Routine Name
Files	
Globals	SETJRN^%GLOMAN
List Journal	^JRNLIS
Operator	

Utility	Routine Name
Add Journal File Name	ADDFIL^JRNL
Close Current Journal File	CLOSEFI^JRNL
Create New Log File	NEWLOG^JRNL
Display Journal File Label	LABEL^JRNL
Display Journal File Names	DISPLAY^JRNL
Initialize Journal File	^JRNINIT
Open Current Journal File	OPENFIL^JRNL

Utility	Routine Name
Disable Journaling	PAUSE^JRNL
Disable Transactions	TPOFF^JRNL
Enable Journaling	RESUME^JRNL
Enable Transactions	TPON^JRNL
Journal Status	STATUS^JRNL

The *VAX DSM Database Operations Guide* describes the After-Image Journaling Utilities in detail.

Before-Image Journaling Utilities

Use the Before-Image Journaling Utilities to create a before-image journal (BIJ) file, examine the contents of the file, and apply the file to repair a corrupted database. You can access the following Before-Image Journaling Utilities from the ^DBSMAN Utility menu:

Utility	Routine Name
Create	^BIJINIT
Examine	^BIJEXAM
Force	^BIJFORC

The *VAX DSM Database Operations Guide* describes the Before-Image Journaling Utilities in detail.

Database Set Utilities

Use the Database Set Utilities to manage database sets. You can access the following utilities from the ^DBSMAN Utility menu:

Utility	Routine Name
Create/Modify Database Set	MODIFY^CONDBS
Delete Database Set	DELETE^CONDBS
Display Database Set	DISPLAY^CONDBS

The Database Set Utilities are described in detail in the *VAX DSM Database Operations Guide*.

Volume Set Utilities (^VOLUTL)

Use the ^VOLUTL Utilities to manage VAX DSM volume sets, and the UCIs and globals associated with them. The ^VOLUTL Utilities also provide backup and restore utilities and database integrity utilities. You can access the following utilities from the ^VOLUTL Utility menu:

Utility	Routine Name
Backup	^BACKUP

Utility	Routine Name
Backup	BCKUP^BACKUP
Restore	REST^BACKUP
Update	REV^BACKUP

Database Repair

Utility	Routine Name
Block Dump	^BLDMP
Fast IC	^FASTIC
Fix	^FIX
Integrity Checker	^IC

Disk Block Tally	^DBT
Global Management	^%GLOMAN
Show Mounted Volume Sets	^%VOLTAB
UCI Management	^UCIMAN

Utility	Routine Name
Add UCI	^UCIADD
Change UCI Library	LIB^UCIEDIT
Edit UCI Name	^UCIEDIT
Show UCIs	SHOW^UCIADD
UCI Growth	^UCIGAM

Utility	Routine Name
Volume Management	^VOLMAN

Utility	Routine Name
Add	ADD^VOLMAN
Create	CREATE^VOLMAN
Extend	EXTEND^VOLMAN
Maximum Globals	MAXGLO^VOLMAN2
String Length	EXPSTR^VOLMAN2

Volume Set Access	^VOLINH
Volume Set Upgrade	^UPGCOM

For information about ^BACKUP, the Database Repair Utilities, and the global management and volume set management utilities, see the *VAX DSM Database Operations Guide*. For information about the ^UPGCOM Utility, see the *VAX DSM Version 6.0 Release Notes*. The following section describes the ^DBT Utility.

Disk Block Tally (^DBT)

The ^DBT Utility reports on the allocation of blocks within a volume set. You have the option to run a fast tally or a slow tally. The fast tally gives you a summary of the map number, the blocks available, and how the blocks are being used. The slow tally gives you a more detailed report.

Distributed Data Processing Utilities (^DDP)

Use the ^DDP Utilities to obtain information about DDP servers and to shut down a server. The ^DDP Utilities include options for controlling DDP operations and displaying information about DDP. You can access the following utilities from the ^DDP Utility menu:

Utility	Routine Name
Circuit Management	^DDPCIR

Utility	Routine Name
Circuit Status	STA^DDPCIR
Disable Circuit	DISAB^DDPCIR
Enable Circuit	ENAB^DDPCIR
Reset Circuit Counters	RESET^DDPCIR
Show Circuit Counters	COUNT^DDPCIR

Configure DDP	^CONDDP
Control Utility	^DDPUTL
Link Status	^DDPLNK

Utility	Routine Name
Security	^DDPSEC

Utility	Routine Name
Add Node Password	ADD^DDPSEC
Delete Node Password	DELETE^DDPSEC
List Database Entries	LIST^DDPSEC
Modify Node Password	MODIFY^DDPSEC

Shutdown DDP	STOP^DDPLNK
Startup DDP	START^DDPLNK
Statistics	^DDPSTA

Utility	Routine Name
Monitor	MONITOR^DDPSTA
Record	RECORD^DDPSTA
Replay	REPLAY^DDPSTA

For information about the DDP Utilities, see the *VAX DSM Database Operations Guide*.

Mapped Routine Utilities (^RMAP)

Use the ^RMAP Utilities to map routines into virtual memory sections to improve system performance. You can access the following utilities from the ^RMAP Utility menu:

Utility	Routine Name
Build	^RMBLD
Contents	^RMSHO
Delete	^RMDEL
Directory	MAP^%RD
Disable Routine	^RMDIS
Install	^RMINS
Show Installed	MAPSET^RMSHO

For information about mapped routines, see Chapter 2 and Chapter 5.

Option File Management Utilities (^OPTMAN)

Use the ^OPTMAN Utility to create a configuration definition file and apply that file to ^CONFIG. You can access the following utilities from the ^OPTMAN Utility menu:

Utility	Routine Name
Convert Option File	^OPTCON

Utility	Routine Name
Create Option File	^OPTBLD

For more information about the ^OPTMAN Utility, see Appendix B.

Performance Utilities

Use the Performance Utilities to gather statistics on DDP use, global and routine performance, and lock management. You can access the following utilities from the System Utilities menu (^SYS):

Utility	Routine Name
DDP Statistics	^DDPSTA
Global Statistics	^GLSSTA

Utility	Routine Name
Monitor	MONITOR^GLSSTA
Record	RECORD^GLSSTA
Replay	REPLAY^GLSSTA

Utility	Routine Name
Lock Management Statistics	^LKMSTA

Utility	Routine Name
Monitor	MONITOR^LKMSTA
Record	RECORD^LKMSTA
Replay	REPLAY^LKMSTA

Utility	Routine Name
Routine and Global Histograms	^RTH

Utility	Routine Name
Help	^RTHELP
Log	^RTHIST
Print	^RTHISTP
Save	SAVE^RTHISTP
Status	STATUS^RTH
Stop	HALT^RTH

Utility	Routine Name
Transaction and Journaling Statistics	^TPSTA

For information on ^GLSSTA and ^RTH, see Chapter 5. For information on ^DDPSTA, ^LKMSTA, and ^TPSTA, see the *VAX DSM Database Operations Guide*.

System Management Utilities (^SYSMAN)

Use the ^SYSMAN Utilities to start up and shut down VAX DSM, to provide access control, and to display memory utilization and system use. You can access the following System Management Utilities from the ^SYSMAN Utility menu:

Utility	Routine Name
Access Control	^ACL
Force Exit	^FORCEX
Lock Table Display	^LOCKTAB
Login	^LOGIN
Memory	^MEMORY
Show DSM Users	^%SY
Shutdown	^SHUTDWN
Startup	^STU
Tied Routines	^TIED

Utility	Routine Name
Add/Modify User	ADD^ACL
Delete User	DELETE^ACL
Modify Active Authorizations	^ACLSET
Print	PRINT^ACL

Utility	Routine Name
Requirement	^MEMREQ
Usage	^MEMUSE

Utility	Routine Name
Add Port	ADD^TTPORT
Delete Port	DELETE^TTPORT
Modify Routine Table	MODIFY^TTROU
Show Tied Ports	SHOW^TTPORT
Show Tied Routines	SHOW^TTROU

For information about the ^ACL, ^LOGIN, ^STU, and ^TIED Utilities, see Chapter 3. For information about the ^MEMORY Utility, see Chapter 4. For information about the ^%SY Utility, see the *VAX DSM Programmer's Guide*.

The following sections describe the ^FORCEX, the ^LOCKTAB, and the ^SHUTDWN Utilities.

Force Exit (^FORCEX)

The ^FORCEX Utility forces image exit for a specified DSM user process. If you want to delete a process running DSM by using the DCL STOP/ID command, you must first run down the image using ^FORCEX. This ensures that the image exits properly from the DSM global subsystem.

Lock Table Display (^LOCKTAB)

The ^LOCKTAB Utility displays the VAX DSM Lock Table. The Lock Table contains the following information:

- Owner PID
- Lock type
- Variable name

Shutdown (^SHUTDWN)

The ^SHUTDWN Utility shuts down a VAX DSM configuration. When you shut down a configuration, background DSM processes are stopped, application processes are exited, and volume sets are dismounted.

Appendix A
Installing VAX DSM Software in an Alternate Root

This appendix describes the procedure for installing VAX DSM in an alternate root.

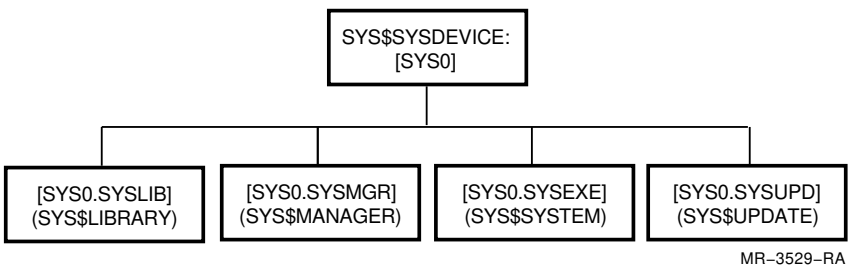
Using VAX DSM from an Alternate System Root

If you want to install a new version of VAX DSM and do not want to overwrite your existing VAX DSM system software, you can install the new release in an alternate system root.

The VMSINSTAL procedure normally copies the various command procedures, routine files, and executable images from the VAX DSM software distribution kit to appropriate *system directories*. System directories are constructed as a hierarchy of directories, which are descendants of a primary system root.

For example, the top level directory of system root SYS0 is SYS$SYSDEVICE:[SYS0]. Figure A–1 describes the system directories that are descendants of the SYS0 system root.

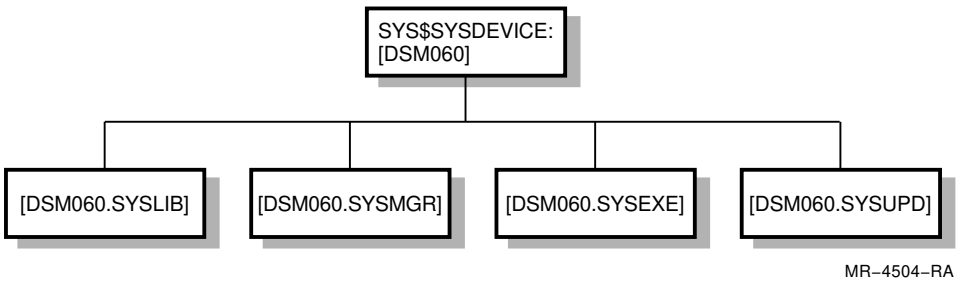
Figure A–1 System Root SYS0 Directory Structure



You can use an option provided by VMSINSTAL to distribute the software kit contents into an alternate root of directories. The alternate root directories are outside of the hierarchy of primary system directories.

An alternate root is a set of directories with a structure that is parallel to the primary system root directories. However, the top level directory of the alternate root is not a VMS system root. For example, an alternate root directory for a special distribution of VAX DSM might be constructed as shown in Figure A–2.

Figure A–2 Alternate Root Directory Structure



When installing the VAX DSM software into this alternate root, files destined for SYS$LIBRARY are deposited by VMSINSTAL into the parallel directory of the alternate root, SYS$SYSDEVICE:[DSM060.SYSLIB]. Files destined for SYS$SYSTEM are deposited in SYS$SYSDEVICE:[DSM060.SYSEXE] and so on.

Once the installation is complete, the alternate root software can be selectively run on a VMS group basis. This is done by redefining system logical names as *search lists* within the target group logical name table.

The following table is an example of how logical name search lists are used to redirect access to DSM software resident in the alternate root:

Logical Name	Search List Equivalent
SYS$LIBRARY	SYS$SYSDEVICE:[DSM060.SYSLIB],SYS$SYSROOT:[SYSLIB]
SYS$MANAGER	SYS$SYSDEVICE:[DSM060.SYSMGR],SYS$SYSROOT:[SYSMGR]
SYS$SYSTEM	SYS$SYSDEVICE:[DSM060.SYSEXE],SYS$SYSROOT:[SYSEXE]

When these search lists are active, an attempt to access a file in SYS$LIBRARY, SYS$MANAGER, or other system directories results in a search for the file in the alternate root first. If the file is not found in the alternate root, the file is accessed from the primary root.

The DSM$ALTROOT.COM procedure initializes the group logical name table. Once the group logical name table is initialized with the search lists, any VMS user logged in to an account within the group runs VAX DSM from the alternate root. Users in other VMS groups continue to run VAX DSM from the primary root software.

Installing software in an alternate root allows you access to the full functionality of the new software without impact to the currently installed version. It provides a safe environment in which you can test and debug until you are satisfied that your application code works under the new release. You can keep one or more releases of VAX DSM on line within separate alternate roots and selectively enable each version for use by specific VMS groups.

Once you are satisfied that you have made all of the required changes to your application, you can reinstall the DSM image into the primary system root and delete the files in the alternate root.

> **Note: You cannot start DDP from an alternate root when an incompatible DDP driver is already loaded into the system.**

The following are the major steps involved in installing software in an alternate root:

1. Use the VMS Backup Utility to extract a copy of the DSM$ALTROOT_INSTALL.COM procedure from the VAX DSM software distribution kit.
2. Use DSM$ALTROOT_INSTALL.COM to create the alternate root directories and install the VAX DSM software distribution kit.
3. Use DSM$ALTROOT.COM to create and initialize a VAX DSM environment manager's account for the new alternate root.
4. Use DSM$ALTROOT.COM to initialize the group logical name search lists for all users within the GROUP.
5. Log in to the new manager account and use the command procedure, SYS$MANAGER:DSM$CONFIGURE.COM, to create volume sets and define new configurations.
6. Edit your system startup files to initialize alternate root installation on system reboot.

Installing VAX DSM in an Alternate Root

Use the following procedure to install VAX DSM software in an alternate VMS root:

1. Log in to the VMS system manager's account.
2. Register and load a valid VAX DSM Version 6.0 license. For detailed information, see Chapter 2 of the *VAX DSM Version 6.0 Release Notes*. Note that the VAX DSM Version 6.0 license that you load is sufficient for operation of VAX DSM Version 6.0 and all previous versions of VAX DSM.

3. Extract the DSM$ALTROOT_INSTALL.COM procedure from the VAX DSM software distribution kit. Enter the following VMS Backup Utility command line to do this:

```
$ BACKUP/LOG KIT_DEVICE:DSM060.B/SAVE -Return
_$ /SELECT=DSM$ALTROOT_INSTALL.COM SYS$MANAGER:*;Return
```

where:

KIT_DEVICE is a tape or disk device on which the DSM kit is located (For example, MUA0. The device can be specified by a logical name.)

4. Install the VAX DSM software into an alternate root using the DSM$ALTROOT_INSTALL.COM procedure by entering the following command line:

```
$ @SYS$MANAGER:DSM$ALTROOT_INSTALL Return
```

The DSM$ALTROOT_INSTALL procedure creates all required alternate root subdirectories. It then automatically runs the SYS$UPDATE:VMSINSTAL procedure with the appropriate options and parameters. The VMSINSTAL software kit installation procedure appears identical to the procedure used for primary system root installations. Primary system root installation is described in detail in Chapter 1.

5. Next, create one or more VAX DSM manager accounts to use with the alternate root installation. You do this by entering the following command line:

```
$ @ddcu:[root_dir]DSM$ALTROOT INIT MANAGER Return
```

where:

ddcu is the physical or logical name of the disk on which you installed the alternate root

root_dir is the top level directory name of the alternate root

6. Finally, use the DSM$ALTROOT command procedure to initialize the group logical name tables for the newly created VAX DSM manager account. Do this by entering the following command:

```
$ @ddcu:[root_dir]DSM$ALTROOT INIT group_number Return
```

where:

ddcu is the physical or logical name of the disk on which you installed the alternate root

root_dir is the top level directory name of the alternate root

group_number is the VMS group number of the account you initialized for use within the alternate root

7. Edit your site-specific VMS startup files so that the VMS groups selected to use the alternate root software are properly initialized at VMS boot time. This is accomplished by calling DSM$ALTROOT once for each group you want to initialize. For example, add the following commands to SYS$STARTUP:SYSTARTUP_V5.COM to initialize groups 210 and 220 for use with alternate root software that is resident in SYS$SYSDEVICE:[DSM060]:

```
$ @SYS$SYSDEVICE:[DSM060]DSM$ALTROOT INIT 210
$ @SYS$SYSDEVICE:[DSM060]DSM$ALTROOT INIT 220
```

   You can also choose to start configurations from the alternate root accounts at VMS boot time. In the following example, DSMMGR1, DSMMGR2, and DSMMGR3 are environment manager accounts that are located in VMS groups 210 and 220 and consequently use alternate root software. Both groups have already been initialized using calls to DSM$ALTROOT. The following commands were added to SYS$STARTUP:SYSTARTUP_V5.COM:

```
$ SUBMIT/USER=DSMMGR1/QUEUE=DSM$BATCH -
                SYS$SYSDEVICE:[DSM060.SYS$STARTUP]DSM$STARTUP
$ SUBMIT/USER=DSMMGR2/QUEUE=DSM$BATCH -
                SYS$SYSDEVICE:[DSM060.SYS$STARTUP]DSM$STARTUP
$ SUBMIT/USER=DSMMGR3/QUEUE=DSM$BATCH -
                SYS$SYSDEVICE:[DSM060.SYS$STARTUP]DSM$STARTUP
```

Running the DSM$ALTROOT_INSTALL.COM Procedure

The following example illustrates the installation of VAX DSM into an alternate root and the initialization of accounts that use the alternate root.

```
$ BACKUP/LOG MUA0:DSM060.B/SAVE -Return
_$ /SELECT=DSM$ALTROOT_INSTALL.COM SYS$MANAGER:*;Return
BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DSM$ALTROOT_INSTALL.COM;1


$ @SYS$MANAGER:DSM$ALTROOT_INSTALLReturn

           VAX DSM Alternate Root Installation Procedure

This procedure will create the directories necessary to
accomplish an alternate root installation of VAX DSM and then
optionally call the VMSINSTAL procedure to do the installation.

Enter the device and root directory for the alternate
root to install into using the format:

ddcu:[DIRECTORY]

where:

ddcu         is the device or logical name of the device
             that will hold the alternate root
[DIRECTORY]  is the root directory name for the alternate
             root (If the directory does not exist, it
             will be created.)
```

```
Enter alternate root device and directory: SYS$SYSDEVICE:[DSM060]Return

Enter alternate root device and directory: SYS$SYSDEVICE:[DSM060]
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSEXE] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSHLP] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSHLP.EXAMPLES] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSLIB] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSMGR] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSMSG] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSUPD] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYS$STARTUP] created
%CREATE-I-CREATED, SYS$SYSDEVICE:[DSM060.SYSTEST] created

The alternate root has been successfully created.
To install VAX DSM into this root, the following command will
be executed:

@SYS$UPDATE:VMSINSTAL DSM KIT_DEVICE OPTIONS R _ _DSM9$DUA0:[DSM060.]

Do you wish to install the VAX DSM kit now [Y or N]: YReturn
Enter the kit device name: MUA0:Return

Installing VAX DSM into root SYS$SYSDEVICE:[DSM060]

            VMS Software Product Installation Procedure 6.0

It is 14-JUN-1989 at 12:32.

Enter a question mark (?) at any time for help.
.
.
.
```

The VMSINSTAL procedure is similar to a primary system root installation at this point. You can answer all questions as though you were installing VAX DSM into a primary system root.

When VMSINSTAL is complete, DSM$ALTROOT_INSTALL displays informational text that describes the steps required to initialize VAX DSM environment manager accounts.

```
.
.
.
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
 directories...

        Installation of DSM V6.0 completed at 12:48

        VMSINSTAL procedure done at 12:48

        Installation of the VAX DSM software kit is now complete.
        To initialize one or more VAX DSM environment manager's
        accounts which use the alternate root software,
        type the following command line while logged into a
        privileged system account:

        @SYS$SYSDEVICE:[DSM060]DSM$ALTROOT INIT MANAGER

Do you want to initialize a manager's account now [Y OR N]:YReturn
```

The installation of the VAX DSM software distribution kit into the alternate root is complete. You must now initialize a manager account for use with the alternate root. Manager account initialization within an alternate root is identical to the process used within a primary system root. However, you do not use the DSM$INIT command procedure to create the manager account. Instead, use the DSM$ALTROOT procedure with the INIT MANAGER parameters to accomplish manager account initialization.

DSM$ALTROOT_INSTALL initializes a manager account on completion of installation if you choose.

```
Installing DSM Alternate Root Images:

          _DSM5$DJA1:[DSM060.SYSEXE]DSM0601.EXE
          _DSM5$DJA1:[DSM060.SYSLIB]DSM$SHARE0601.EXE
          _DSM5$DJA1:[DSM060.SYSMSG]DSM$MESSAGE0601.EXE
          _DSM5$DJA1:[DSM060.SYSLIB]DSM$ZCALL0601.EXE

Initializing Logical name table:     LNM$PROCESS_TABLE


                        VAX DSM V6.0
                  Environment Manager Initialization

  The VAX DSM environment manager must have a VMS account
  to define and control VAX DSM configurations.  This procedure
  allows you to create a new account or upgrade an existing
  VMS account for use as a VAX DSM environment manager.


Enter the VMS account name to use with VAX DSM  <DSMMGR>DSMALT Return

The DSMALT account does not currently exist
Do you wish to create and initialize this account [Y OR N] ? <Y> Return


  Each VMS account must have a UIC identifier associated with it.
  A UIC is in the form:

      [GROUP,MEMBER]

  Where GROUP is an octal number less than  37777
  and  MEMBER is an octal number less than 177777

  Enter the UIC to apply to this account when it is created.

Enter the UIC for this account   >[210,1] Return
.
.
.
```

Initialization of alternate root software is done on a VMS group basis, so when you create an account for use with alternate root software, choose a VMS group that does not have users currently using VAX DSM software within the primary root.

The remainder of the manager account initialization process for alternate root software is identical to the process used for primary root installations. Again, answer the remaining questions as though you were initializing an account for a primary root installation. For more detailed information, see Chapter 1.

When DSM$ALTROOT completes, it describes the command required to set up the group logical name table for use with the alternate root.

```
.
.
.
The DSM management globals will now be updated . . .

Updating DSM Management Global:         ^[SYS$LIBRARY]DSMMGR
Initializing DSM Configuration Global:  ^[DISK$USER:[DSMALT]]SYS

Account DSMALT successfully initialized

The account has been created with your specified password.
However, for security reasons, the password has been set to expire.
Please log in to the new account immediately and reset the account
password.

You may now log in to the VAX DSM manager's account and
use the SYS$MANAGER:DSM$CONFIGURE.COM command procedure to
initialize VAX DSM database volume sets, or create and start
up VAX DSM configurations.

See the VAX DSM Installation and Management Guide for complete
information about VAX DSM database volume sets and configurations.

Before logging into the environment manager account, you must enter
the following command from a privileged account:

@SYS$SYSDEVICE:[DSM060]DSM$ALTROOT INIT 210

Note that this command must be reentered each time you reboot
your VAX VMS system in order to install the DSM alternate root
images and create appropriate VMS group logical names.
```

The manager's account is initialized. You can initialize other environment manager accounts at a later time if necessary. To do so, enter the command:

```
$ @SYS$SYSDEVICE:[DSM060]DSM$ALTROOT INIT MANAGER Return
```

To prepare a VMS group for use of the alternate software, you must call the DSM$ALTROOT procedure again. This time the group number of the VAX DSM environment manager's account just initialized is supplied as the input parameter. The DSM$ALTROOT procedure installs the VAX DSM alternate root images as shareable, resident images using the VMS Install Utility. It creates the appropriate logical name search lists for the group being initialized.

The following dialogue is an example of how to use DSM$ALTROOT:

```
$ @SYS$SYSDEVICE:[DSM060]DSM$ALTROOT INIT 210 Return
Installing DSM Alternate Root Images:

          _DSM5$DJA1:[DSM060.SYSEXE]DSM0601.EXE
          _DSM5$DJA1:[DSM060.SYSLIB]DSM$SHARE0601.EXE
          _DSM5$DJA1:[DSM060.SYSMSG]DSM$MESSAGE0601.EXE
          _DSM5$DJA1:[DSM060.SYSLIB]DSM$ZCALL0601.EXE

Initializing Logical name table:     LNM$GROUP_000210
Alternate root successfully initialized for VMS group 000210
```

You can now log in to the VAX DSM manager's account (DSMALT) and run VAX DSM Version 6.0 utilities and procedures just as though you were running from a primary system root.

Installing VAX DSM in an Alternate Root in a VAXcluster Environment

If you are installing VAX DSM in a VAXcluster environment, you must log in to the VMS system manager's account on each node that will be running VAX DSM and do the following:

```
$ @SYS$DEVICE:[DSM060]DSM$ALTROOT INIT 210 Return
```

This allows you to run VAX DSM on that node in the cluster.

Shutting Down VAX DSM Configurations that Use Alternate Root Software

When you shut down your VMS system, ensure that all VAX DSM configurations are shut down in an orderly manner. You can use the DSM$ALTROOT command procedure to shut down configurations that use alternate root software.

To shut down a configuration running in an alternate root, include the following command line for each configuration in the SYS$MANAGER:SYSHUTDOWN.COM procedure :

```
$ @ddcu:[root_dir]DSM$ALTROOT SHUTDOWN envnm
```

where:

ddcu	is the physical or logical name of the disk on which you installed the alternate root
root_dir	is the top level directory name of the alternate root
envnm	is the VMS user name of the VAX DSM environment manager account

Appendix B
Using the Configuration Option File Utility (^OPTMAN)

This appendix describes the Configuration Option File Utility (^OPTMAN), which lets you create a sequential file containing data keywords, data definitions, and values that define a configuration. The ^STU Utility accepts input from this file.

Using the Configuration Option File

The configuration option file is a VMS sequential file that defines a VAX DSM configuration using *definition keywords* and *data keywords*. It allows the VAX DSM manager to redefine a configuration on the target system without using the ^CONFIG Utility.

The definition keywords are equivalent to the ^CONFIG questions. Data keywords are associated with definition keywords by function. Values are assigned to the data keywords to form the configuration definition. For example, DEFINE DDP is a definition keyword and NODE_NAME= is a data keyword associated with DEFINE DDP. You assign a value to the data keyword (for example, NODE_NAME="ABC").

You create a configuration option file with a data value defined for each option just as you would answer questions in the ^CONFIG Utility. Default data values are assumed for those options that are not present in the option file.

The following sections describe how to create an option file and how to use the option file to start a configuration. The definition keywords and data keywords are described in Table B–1 and Table B–2. A DSM configuration option file example is shown in Example B–1.

Configuration Option File Utilities (^OPTMAN)

You can use the ^OPTMAN Utility to create a configuration definition file and apply that file to ^CONFIG. You can use the Display Configuration Utility (DISPLAY^CONFIG) to display the characteristics of a defined configuration.

Enter the following to run the ^OPTMAN Utility:

```
>D ^OPTMAN Return
```

The utility displays the following menu:

```
Option File Management Utilities

  1.  CONVERT OPTION FILE           (^OPTCON)
  2.  CREATE OPTION FILE            (^OPTBLD)

Select Option >
```

The following sections describe these utilities.

Creating an Option File (^OPTBLD)

Use the Create Option File Utility (^OPTBLD) to create a configuration option file from an **existing** configuration defined by the ^CONFIG Utility. You can access ^OPTBLD from the System Utilities menu (^SYS), or you can enter the following:

```
>D ^OPTBLD Return


     Create Configuration Option File


Enter the configuration identifier  <PAYROLL> Return

Enter the name of the configuration for which you want to create
an option file. The name you enter must be a currently defined
configuration. Press Return to accept the default or enter a
circumflex (^) to exit. Enter ^L to display a list of the
defined configurations.


File name of option file <DISK$USER:[DSM]DSM$CONFIG_PAYROLL.OPT> Return

Enter the VMS file specification of the configuration option file
to be created. You can specify a logical name that translates to
a complete file specification.


Creating option file DISK$USER:[DSM]DSM$CONFIG_PAYROLL.OPT;1 ...

Option file DISK$USER:[DSM]DSM$CONFIG_PAYROLL.OPT;1
has been successfully created
```

You can use any VMS editor to modify the contents of the option file. The recommended procedure is to generate a template option file using the ^OPTBLD Utility and then edit the configuration data to create an option file that specifies the new configuration you plan to use.

Converting the Option File (^OPTCON)

Use the Convert Option File Utility (^OPTCON) to convert the configuration option file created by ^OPTBLD to the configuration database that is used to start up a DSM configuration. You can access ^OPTCON from the System Utilities menu (^SYS), or enter the following:

```
>D ^OPTCON Return


     Convert Configuration Option File


Enter the configuration identifier <PAYROLL> Return

Enter the name of the configuration identifier for which you want
to convert an option file or press Return to accept the default.
Enter a circumflex (^) to exit from the routine. Enter ^L to
display a list of defined configurations.


File name of option file<DISK$USER:[DSM]DSM$CONFIG_PAYROLL.OPT> Return

Enter a valid VMS file specification for the configuration
option file. The option file is used to build a configuration
database for starting up a DSM configuration. You can specify
a logical name that translates to a complete specification.


Building VAX DSM startup options ...

Option file DISK$USER:[DSM]DSM$CONFIG_PAYROLL.OPT;1 processing complete
^[DISK$USER:[DSM]]SYS("PAYROLL") created
```

In the previous example, the ^OPTCON Utility creates a configuration database and assigns PAYROLL as the configuration ID. Each time you convert a new option file, the old configuration information is automatically updated in the database.

Starting Up a Configuration

You can start a DSM configuration from the configuration option file by doing one of the following:

- Use the ^OPTCON Utility to build a configuration database and start up the configuration using the ^STU Utility. For example:

```
>D ^STU Return

     DSM Configuration Startup


Enter the configuration identifier  <PAYROLL> Return
```

  In the previous example, PAYROLL is the configuration ID created by the ^OPTCON Utility.

- Use the OPT^STU Utility to start up the configuration. You must run the utility from the DCL command line using the DSM command. The OPT^STU entry point accepts two arguments (separated by a comma) for input data. The following is the format of the command line:

```
$ DSM/MANAGER/DATA=arg1,arg2 OPT^STU
```

  where:

  arg1 — is the name of a configuration option file

  arg2 — is ENABLE to enable user logins on startup, and DISABLE to disable user logins on startup

  For example, enter the following:

```
$ DSM/MANAGER/DATA="DSM$:[DSM]DSM$CONFIG_PAYROLL.OPT,ENABLE" OPT^STU Return
```

  When you enter an existing option file specification, the OPT^STU Utility automatically converts the file to the configuration database and starts up the DSM configuration.

Configuration Option File Keywords

The configuration option file contains two types of keywords:

- Definition keywords
- Data keywords

Definition keywords separate the data keywords into different functional groups and data keywords define the data value for a startup option.

The ^OPTCON Utility aborts any attempt to convert an option file into the configuration database if one or more incorrect entries are present in the file.

Do the following when you create an option file for a configuration:

- Use definition keywords to define a group of data keywords that are placed under the same category and perform one startup function for a configuration, such as mapped routines or Distributed Data Processing (DDP).
- Use data keywords to assign data values to a configuration startup option such as the number of disk buffers, or the file name of a DSM volume.
- Data keywords may or may not require a definition keyword. Use Table B–1 to look up the definition keywords and use Table B–2 to place the data keywords under the appropriate definition keyword.
- Use data keywords and definition keywords to define startup options for a configuration. Default values are automatically assigned to those options that are not defined in the configuration option file.
- All data values assigned to the data keywords are validated by the ^CONFIG Utility.
- Each data keyword must be followed by an equals sign (=) and a data value. Leading and trailing space characters are ignored.
- The data value must be enclosed by a pair of identical ASCII punctuation characters, such as /300/ or "300".
- Comment lines must beging with a slash character (/) or an exclamation mark (!). You can place comment lines in the configuration option file for documentation purposes.
- Blank lines are not allowed in the configuration option file.

Table B–1 describes the definition keywords.

Table B–1 Definition Keywords

Keyword and Description	Example
DEFINE DATABASE_SET	
Specifies the configuration startup options for a VAX DSM database set.	DEFINE DATABASE_SET VOLUME_SET_MASTER="Y" CLUSTER_WIDE_ACCESS="N" DATABASE_SET_NAME="PAYROLL" DATABASE_SET_TYPE="VAX DSM"
DEFINE VOLUME_SET	
Specifies the configuration startup options for a volume set. This keyword must appear under the DEFINE DATABASE_SET definition keyword.	DEFINE VOLUME_SET VOLUME_SET_NAME="EMP" ALTERNATE_NAME="REC"
DEFINE VOLUME	
Specifies the configuration startup options for a VAX DSM volume. There must be a DEFINE VOLUME section for each volume set that was created. This keyword must appear under the DEFINE VOLUME_SET definition keyword.	DEFINE VOLUME FILE_NAME="DISK$USER:[DSM]EMPVOL1.GLS" DEFINE VOLUME FILE_NAME="DISK$USER:[DSM]EMPVOL2.GLS"
DEFINE AIJ	
Specifies the startup options for the After-Image Journaling file. This keyword must appear under the DEFINE DATABASE_SET definition keyword.	DEFINE AIJ FILE_NAME="DISK$USER:[DSM]EMPJRN1.DAT" MESSAGE="THIS IS FOR AIJ" REPLY="N" DEFINE AIJ FILE_NAME="DISK$USER:[DSM]EMPJRN2.DAT" MESSAGE="MORE AFTER-IMAGE JOURNALING" REPLY="N"
DEFINE BIJ	
Specifies the startup options for the Before-Image Journaling file. This keyword must appear under the DEFINE VOLUME_SET definition keyword.	DEFINE BIJ FILE_NAME="DISK$USER:[DSM]DSMEMP.BIJ"

Table B–1 (Cont.) Definition Keywords

Keyword and Description	Example
DEFINE DDP	
Specifies the configuration startup options for Distributed Data Processing (DDP).	DEFINE DDP NODE_NAME="ABC" STARTUP="Y" SERVERS="3" OPCOM="Y" GROUPS="0" RETRY="4" RETRY_INTERNAL="3" SECURITY="N" REQUEST_CREDIT="20" TRANSMIT_DELAY="2" ADVERTISING_INTERVAL="6"
DEFINE DDP_LINK	
Specifies the configuration startup options for the DDP link type and device type. Note, this keyword must appear under the DEFINE DDP definition keyword.	DEFINE DDP_LINK LINK_TYPE="ETHERNET" LINK_DEVICE="XEA:"
DEFINE HISTOGRAM	
Specifies the configuration startup options for routine and global performance histograms.	DEFINE HISTOGRAM BUFFER_SIZE="50"
DEFINE MAPPED_ROUTINES	
Specifies the configuration startup options for a mapped routine set.	DEFINE MAPPED_ROUTINES UCI_NAME="MGR" VOLUME_SET_NAME="EMP" DIRECTORY_NAME="DISK$USER:[DSM]" SECTION_TYPE="P" PAGE_CLUSTER_SIZE="8"
DEFINE TRANSLATION	
Specifies an entry in the Translation Table for a configuration.	DEFINE TRANSLATION GLOBAL_NAME="TEST" OLD_UCI_VOL="MGR,EMP" NEW_UCI_VOL="MGR,TST" 8_BIT_ENCODING="Y" GLOBAL_COLLATING="NUMERIC"
DEFINE SCHEMA	
Specifies a new schema in the Schema Table for a configuration.	DEFINE SCHEMA LOCK_UCI_VOL="MGR,EMP" READ_UCI_VOL="MGR,EMP" MODIFY_UCI_VOL="MGR,TST/MGR,DEF"

Table B–1 (Cont.) Definition Keywords

Keyword and Description	Example
DEFINE GLOBAL_DEFAULT	
Specifies the configuration startup options for default DSM global characteristics.	DEFINE GLOBAL_DEFAULT 8_BIT_ENCODING="Y" GLOBAL_COLLATING="N" GLOBAL_JOURNAL="Y"
DEFINE PROTECTION	
Specifies the configuration startup options for global protection defaults. It must appear under the DEFINE GLOBAL_DEFAULT definition keyword.	DEFINE PROTECTION %_GLOBAL_SYSTEM="RWD" %_GLOBAL_WORLD="R" %_GLOBAL_GROUP="R" %_GLOBAL_UCI="RWD" GLOBAL_SYSTEM="RWD" GLOBAL_WORLD="NONE" GLOBAL_GROUP="NONE" GLOBAL_UCI="RWD"
DEFINE AIJ_DEFAULT	
Specifies the configuration startup options for default After-Image Journaling characteristics.	DEFINE AIJ_DEFAULT BUFFER_SIZE="8" OPERATOR_CONSOLE="CENTRL" LOG_FILE_DIRECTORY="DISK$USER:[DSM]"
DEFINE TRANSACTION_PROCESSING	
Specifies the configuration startup options for transaction processing (TP).	DEFINE TRANSACTION_PROCESSING MAXIMUM_TRANSACTIONS="12" TP_SINGLE_REFERENCE="Y" REDO_BUFFERS="64"
DEFINE TP_MONITOR	
Specifies the configuration startup options for the transaction processing monitor. It must appear under the DEFINE TRANSACTION_ PROCESSING definition keyword.	DEFINE TP_MONITOR CHECKPOINT_TIMER="180" CLUSTER_TRANSACTIONS="20" INTERLOCK_TABLE_SIZE="96"

Table B–1 (Cont.) Definition Keywords

Keyword and Description	Example
DEFINE LOGIN_DEFAULT	
Specifies the qualifiers you want to append to the DSM command line whenever the DSM image is run.	DEFINE LOGIN_DEFAULT UCI_NAME="DEF" VOLUME_SET_NAME="EMP" ROUTINES="[DEF,EMP]" SOURCE_BUFFER_SIZE="2048" SYMBOL_TABLE_SIZE="1024" TERMINAL_BUFFER_SIZE="1024" STACK_SIZE="4096" MAPPED="Y" TYPEAHEAD="N" PACK="N" PASSALL="N" BYPASS="N" CENABLE="N" DPRECISION="Y" NOUNDEFINED="Y" SETFUNCTION="N"

Table B–2 describes each of the data keywords.

Table B–2 Data Keywords

Keyword and Description	Example	Required Definition Keyword
DISK_BUFFERS		
The number of disk buffers used to cache database disk data in memory.	DISK_BUFFERS="300"	None
NON_PAGED_MEMORY		
Use nonpaged memory to contain the disk buffers that are used for shared global access.	NON_PAGED_MEMORY="N"	None
USERS		
The maximum number of DSM users you want to allow simultaneous access to this configuration.	USERS="128"	None
DATABASE_SET_NAME		
Name of a database set to mount during configuration startup.	DATABASE_SET_NAME="PAYROLL"	DEFINE DATABASE_SET
DATABASE_SET_TYPE		
Specifies the database set as one of three types of database sets: VAX DSM, DSM-11 compatible, or Transaction Processing.	DATABASE_SET_TYPE="VAX DSM"	DEFINE DATABASE_SET
GLOBAL_NAME		
Name of a DSM global. Global name can range from one to eight characters. The first character must be either a letter or the percent character (%).	GLOBAL_NAME="TEST"	DEFINE TRANSLATION
OLD_UCI_VOL		
The global reference is translated from the UCI,VOL pair you specified.	OLD_UCI_VOL="MGR,EMP"	DEFINE TRANSLATION

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
NEW_UCI_VOL		
The global reference is translated to the UCI,VOL pair you specified.	NEW_UCI_VOL="MGR,TST"	DEFINE TRANSLATION
LOCK_UCI_VOL		
The LOCK operations are translated to the UCI,VOL pair you specified. A LOCK operation refers to any reference that performs a global LOCK.	LOCK_UCI_VOL="MGR,EMP"	DEFINE SCHEMA
READ_UCI_VOL		
The read operations are translated to the UCI,VOL pair you specified. A READ operation refers to any reference that performs a global get such as $DATA, $ORDER, $QUERY, or SET variable=^ABC.	READ_UCI_VOL="MGR,EMP"	DEFINE SCHEMA
MODIFY_UCI_VOL		
The modify operations are translated to the UCI,VOL pairs you specified. Up to 29 UCI,VOL pairs, separated by a / character, can be entered. A MODIFY operation refers to any reference that performs a global modification such as SET or KILL.	MODIFY_UCI_VOL="MGR,TST/MGR,DEF"	DEFINE SCHEMA
VOLUME_SET_NAME		
The name of the volume set to be used for a configuration startup option. DSM volume set names must be 3-uppercase alphabetic characters.	VOLUME_SET_NAME="EMP"	DEFINE VOLUME_SET DEFINE MAPPED_ROUTINES DEFINE LOGIN_DEFAULT
CLUSTER_WIDE_ACCESS		
The volume set is to be mounted by two or more concurrently running configurations within a VAXcluster system.	CLUSTER_WIDE_ACCESS="N"	DEFINE DATABASE_SET

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
VOLUME_SET_MASTER		
The configuration is selected as the clusterwide volume set master for this volume set if you are using After-Image Journaling on a cluster-mounted volume set.	VOLUME_SET_MASTER="Y"	DEFINE DATABASE_SET
FILE_NAME		
The VMS file specification of a DSM volume or RMS file. The file name can be a VMS logical name that translates to a complete specification.	FILE_NAME= "DISK$USER:[DSM]EMPVOL1.GLS"	DEFINE VOLUME DEFINE AIJ DEFINE BIJ
MESSAGE		
The text of a message you want to transmit to an operator when the journal file is filled and journaling to the next file is ready to begin.	MESSAGE="THIS IS A TEST"	DEFINE AIJ
REPLY		
Demand an operator reply to the Journal Process before allowing the Journal Process to go on to the next journal file.	REPLY="Y"	DEFINE AIJ
NODE_NAME		
Specifies a unique 3-character node name for this DDP node. Each configuration within a DDP network must have a unique node name.	NODE_NAME="ABC"	DEFINE DDP
STARTUP		
Start DDP servers during DSM configuration startup.	STARTUP="Y"	DEFINE DDP
SERVERS		
Specifies the number of DDP servers that are responsible for servicing DDP requests for remote users.	SERVERS="3"	DEFINE DDP

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
OPCOM		
Send DDP circuit and status messages to the operator's console.	OPCOM="Y"	DEFINE DDP
GROUPS		
Specifies the DDP groups to enable for this node. DDP connections are established only between nodes that belong to the same group.	GROUPS="0"	DEFINE DDP
RETRY		
The DDP request retry maximum mechanism is used to recover from instances where a DDP request is not answered by the remote node. The request is resent every 30 seconds until the request is answered, or the request retry maximum is exceeded.	RETRY="4"	DEFINE DDP
RETRY_INTERVAL		
The DDP request retry interval is the time that DDP waits for a response to a DDP request. When the request retry interval expires, DDP resends the DDP request unless the request retry maximum has been exceeded.	RETRY_INTERVAL="3"	DEFINE DDP
SECURITY		
The DDP circuit connection security facility allows the user to control which remote nodes on the network are permitted to connect to this node. Control is enforced through the use of DDP node passwords.	SECURITY="N"	DEFINE DDP
REQUEST_CREDIT		
The DDP request credit maximum is used for buffer flow control of DDP receive buffers.	REQUEST_CREDIT="20"	DEFINE DDP

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
TRANSMIT_DELAY		
The DDP transmit buffer delay increment is a mechanism that delays DDP message transmission in an attempt to allow multiple messages to be packed into a single DDP buffer.	TRANSMIT_DELAY="0"	DEFINE DDP
ADVERTISING_INTERVAL		
The multicast advertising interval is used by the DDP servers to send control messages periodically to all remote DDP nodes to communicate status information.	ADVERTISING_INTERVAL="6"	DEFINE DDP
LINK_TYPE		
Configure an Ethernet type link or a DECnet type link for DDP.	LINK_TYPE="ETHERNET"	DEFINE DDP_LINK
LINK_DEVICE		
The VMS device name of the Ethernet controller that is used for DDP communications.	LINK_DEVICE="XEA:"	DEFINE DDP_LINK
DECNET_CHANNEL		
The DECnet channel specification for DECnet type DDP links.	DECNET_CHANNEL= "DSM::"TASK=DSM$DDPEMP""	DEFINE DDP
BUFFER_SIZE		
The journal buffer size in 1KB VAX DSM blocks or the size in pages of the RTHIST buffer to reserve.	BUFFER_SIZE="8"	DEFINE AIJ_DEFAULT DEFINE HISTOGRAM
UCI_NAME		
Specifies the 3-character UCI name for a volume set.	UCI_NAME="MGR"	DEFINE MAPPED_ROUTINES DEFINE LOGIN_DEFAULT
DIRECTORY_NAME		
Specifies the VMS global directory name for a mapped routine set.	DIRECTORY_NAME="DISK$USER:[DSM]"	DEFINE MAPPED_ROUTINES

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
SECTION_TYPE		
Use nonpaged memory for a mapped routine section.	SECTION_TYPE="P"	DEFINE MAPPED_ROUTINES
PAGE_CLUSTER_SIZE		
The page cluster-size is the number of 512 byte memory pages that are paged in when access to a routine within the routine section causes a page fault.	PAGE_CLUSTER_SIZE="8"	DEFINE MAPPED_ROUTINES
ENABLE_ACCESS_CONTROL		
Enable configuration access control to provide a complete security system for a running configuration. Is you enable access control, you can use the ^ACL Utility to specify which VMS users are permitted to log in to VAX DSM for each of the access modes (Manager, Programmer, or Application).	ENABLE_ACCESS_CONTROL="Y"	None
8_BIT_ENCODING		
Select the 8-bit format subscript as default global encoding scheme.	8_BIT_ENCODING="Y"	DEFINE GLOBAL_DEFAULT
GLOBAL_COLLATING		
Select string or numeric collating as the default collating sequence.	GLOBAL_COLLATING="N"	DEFINE GLOBAL_DEFAULT
GLOBAL_JOURNAL		
Select globals to be journaled by default.	GLOBAL_JOURNAL="Y"	DEFINE GLOBAL_DEFAULT
%_GLOBAL_SYSTEM		
Select % global protection system default.	%_GLOBAL_SYSTEM="RWD"	DEFINE PROTECTION
%_GLOBAL_WORLD		
Select % global protection world default.	%_GLOBAL_WORLD="R"	DEFINE PROTECTION

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
%_GLOBAL_GROUP		
Select % global protection group default.	%_GLOBAL_GROUP="R"	DEFINE PROTECTION
%_GLOBAL_UCI		
Select % global protection UCI default.	%_GLOBAL_UCI="RWD"	DEFINE PROTECTION
GLOBAL_SYSTEM		
Select non-% global protection system default.	GLOBAL_SYSTEM="RWD"	DEFINE PROTECTION
GLOBAL_WORLD		
Select non-% global protection world default.	GLOBAL_WORLD="NONE"	DEFINE PROTECTION
GLOBAL_GROUP		
Select non-% global protection group default.	GLOBAL_GROUP="NONE"	DEFINE PROTECTION
GLOBAL_UCI		
Select non-% global protection UCI default.	GLOBAL_UCI="RWD"	DEFINE PROTECTION
OPERATOR_CONSOLE		
Specifies the name of the operator console to receive journal process messages sent when a journal file is full and an operator reply is needed.	OPERATOR_CONSOLE="CENTRL"	DEFINE AIJ_DEFAULT
LOG_FILE_DIRECTORY		
Specifies the name of the VMS device and directory that holds the journal log files.	LOG_FILE_DIRECTORY= "DISK$USER:[DSM.JOURNAL]"	DEFINE AIJ_DEFAULT

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
TP_SINGLE_REFERENCE		
Specifies how non-transacted references to the transaction database set should be handled. Enter Y if non-transacted references should return an error.	TP_SINGLE_REFERENCE="Y"	DEFINE TRANSACTION_ PROCESSING
MAXIMUM_TRANSACTIONS		
The maximum number of concurrent transactions you want to allow for the node on which this configuration will be started. A total of 128 transactions can be open at one time.	MAXIMUM_TRANSACTIONS="12"	DEFINE TRANSACTION_ PROCESSING
REDO_BUFFERS		
The number of 1KB REDO BUFFERS to be allocated on this node for use by the transaction processor. An insufficient number of REDO BUFFERS results in excessive writing to the Before-Image Journal (BIJ) File and causes excessive checkpoint operations.	REDO_BUFFERS="64"	DEFINE TRANSACTION_ PROCESSING
CHECKPOINT_TIMER		
The number of seconds between transaction processing system checkpoint events. It defines the maximum amount of data that needs to be recovered in a failure event such as a transaction abort or node failure.	CHECKPOINT_TIMER="180"	DEFINE TP_MONITOR
CLUSTER_TRANSACTIONS		
The maximum number of concurrent transactions that can be supported by the transaction processing monitor. It defines the number of concurrent transactions allowed in the cluster at any one time.	CLUSTER_TRANSACTIONS="20"	DEFINE TP_MONITOR

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
INTERLOCK_TABLE_SIZE		
The number of entries in the transaction processing monitor's interlock table. This table is used to keep track of the interdependencies between concurrent transactions.	INTERLOCK_TABLE_SIZE="96"	DEFINE TP_MONITOR
ROUTINES		
The /ROUTINES=[UCI,VOL] command line qualifier selects a nondefault routine directory. This qualifier allows you to separate the UCI and volume set that your globals are located in from the UCI and volume set that your routines are located in.	ROUTINES="[DEF,EMP]"	DEFINE LOGIN_DEFAULT
SOURCE_BUFFER_SIZE		
The /SOURCE_BUFFER_SIZE=n command line qualifier establishes the size, in bytes, of the DSM source routine buffer and precompiled routine buffer.	SOURCE_BUFFER_SIZE="2048"	DEFINE LOGIN_DEFAULT
SYMBOL_TABLE_SIZE		
The /SYMBOL_TABLE_SIZE=n command line qualifier changes the size of your local DSM symbol table to the value specified.	SYMBOL_TABLE_SIZE="1024"	DEFINE LOGIN_DEFAULT
TERMINAL_BUFFER_SIZE		
The /TERMINAL_BUFFER_ SIZE=n command line qualifier changes the size of your terminal's output buffer to the value specified. The size of this buffer determines the maximum size of a single physical write operation to the terminal.	TERMINAL_BUFFER_SIZE="1024"	DEFINE LOGIN_DEFAULT
STACK_SIZE		
The /STACK_SIZE=n command line qualifier changes the size of the DSM call stack to the size specified.	STACK_SIZE="4096"	DEFINE LOGIN_DEFAULT

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
MAPPED		
The /MAPPED command line qualifier establishes access to a set of DSM routines that are mapped in a global virtual memory section.	MAPPED="Y"	DEFINE LOGIN_DEFAULT
TYPEAHEAD		
The /TYPEAHEAD command line qualifier overrides the DSM default by which the READ literal command (READ "Value: ",X) causes the type-ahead buffer to be purged. /TYPEAHEAD always allows type-ahead regardless of the form of the MUMPS READ command.	TYPEAHEAD="N"	DEFINE LOGIN_DEFAULT
PACK		
The /[NO]PACK command line qualifier applies only to terminal output. When /PACK is used, VAX DSM does not perform a $QIO write for each MUMPS WRITE command that is executed. Instead, VAX DSM packs output strings into the terminal output buffer until the buffer is full, or until VAX DSM executes a command that requires flushing the output buffer.	PACK="N"	DEFINE LOGIN_DEFAULT
PASSALL		
The /PASSALL command line qualifier causes VAX DSM to use unformatted output (IO$M_NOFORMAT) when writing to a terminal.	PASSALL="N"	DEFINE LOGIN_DEFAULT
BYPASS		
The /BYPASS command line qualifier disables protection verification on volume set globals. /NOBYPASS enables protection verification, in which case you must run ^%GLOMAN to modify protection on a global by global basis.	BYPASS="N"	DEFINE LOGIN_DEFAULT

Table B–2 (Cont.) Data Keywords

Keyword and Description	Example	Required Definition Keyword
CENABLE		
The /CENABLE command line qualifier enables Ctrl/c recognition. /NOCENABLE disables Ctrl/c recognition.	CENABLE="N"	DEFINE LOGIN_DEFAULT
DPRECISION		
The /EXTENSION=DPRECISION command line qualifier enables extended (double) precision math.	DPRECISION="N"	DEFINE LOGIN_DEFAULT
NOUNDEFINED		
The /EXTENSION=NOUNDEFINED command line qualifier forces VAX DSM to return a null string on any attempt to retrieve the value of an undefined variable.	NOUNDEFINED="N"	DEFINE LOGIN_DEFAULT
SETFUNCTION		

The /EXTENSION=SETFUNCTION command line qualifier enables the following non-ANSI standard forms of the SET command:
`SET $EXTRACT(string,start,end)=expression`
`SET $FIND(string,substring,position)=expression`
	SETFUNCTION="N"	DEFINE LOGIN_DEFAULT

The following example shows the minimum entries that need to be in the configuration option file.

```
!
!   VAX DSM configuration option file
!
!   Name of file:      DISK$USER:[VAXDSM]DSM$CONFIG_PAYROLL.OPT;1
!   Configuration ID:  PAYROLL
!   Date of creation:  26-APR-1990 15:15:43.75
!
DISK_BUFFERS="300"
DEFINE DATABASE_SET
       DATABASE_SET_NAME="PAYROLL"
       DEFINE VOLUME_SET
       VOLUME_SET_NAME="EMP"
              DEFINE VOLUME
                     FILE_NAME="DISK$USER:[VAXDSM.DATABASE]EMPVOL1.GLS"
```

Example B–1 shows a sample of what your configuration option file can look like.

Example B–1 Sample Configuration Option File

```
!
!   VAX DSM configuration option file
!
!   Name of file:      DISK$USER:[VAXDSM]DSM$CONFIG_PAYROLL.OPT;1
!   Configuration ID:  PAYROLL
!   Date of creation:  27-APR-1990 10:32:03.52
!
ENABLE_ACCESS_CONTROL="Y"
DEFINE AIJ_DEFAULT
       BUFFER_SIZE="8"
       LOG_FILE_DIRECTORY="DISK$USER:[VAXDSM.JOURNAL]"
       OPERATOR_CONSOLE="CENTRL"
DISK_BUFFERS="300"
DEFINE DATABASE_SET
       DATABASE_SET_NAME="PAYROLL1"
       DATABASE_SET_TYPE="TP SET"
       CLUSTER_WIDE_ACCESS="Y"
       VOLUME_SET_MASTER="Y"
       DEFINE AIJ
              FILE_NAME="DISK$USER:[VAXDSM.JOURNAL]EMPJRN1.DAT"
       DEFINE AIJ
              FILE_NAME="DISK$USER:[VAXDSM.JOURNAL]EMPJRN2.DAT"
              MESSAGE="THIS IS A TEST"
              REPLY="Y"
       DEFINE VOLUME_SET
       VOLUME_SET_NAME="EMP"
              DEFINE BIJ
                     FILE_NAME="DISK$USER:[VAXDSM.JOURNAL]DSMEMP.BIJ"
              DEFINE VOLUME
                     FILE_NAME="DISK$USER:[VAXDSM.DATABASE]EMPVOL1.GLS"
              DEFINE VOLUME
                     FILE_NAME="DISK$USER:[VAXDSM.DATABASE]EMPVOL2.GLS"
DEFINE DATABASE_SET
       DATABASE_SET_NAME="PAYROLL2"
       DATABASE_SET_TYPE="VAX DSM"
       CLUSTER_WIDE_ACCESS="N"
       VOLUME_SET_MASTER="Y"
       DEFINE VOLUME_SET
       VOLUME_SET_NAME="TST"
              DEFINE VOLUME
                     FILE_NAME="DISK$USER:[VAXDSM.DATABASE]TSTVOL1.GLS"
```

Example B–1 Cont'd on next page

Example B–1 (Cont.) Sample Configuration Option File

```
DEFINE DDP
       ADVERTISING_INTERVAL="6"
       GROUPS="0"
       DEFINE DDP_LINK
              LINK_DEVICE="XEA:"
              LINK_TYPE="ETHERNET"
       REQUEST_CREDIT="20"
       RETRY="4"
       NODE_NAME="ABC"
       OPCOM="Y"
       RETRY_INTERVAL="3"
       SECURITY="N"
       SERVERS="3"
       STARTUP="Y"
       TRANSMIT_DELAY="0"
DEFINE LOGIN_DEFAULT
       CENABLE="Y"
       MAPPED="N"
       PACK="Y"
       PASSALL="Y"
       SOURCE_BUFFER_SIZE="20480"
       STACK_SIZE="4096"
       SYMBOL_TABLE_SIZE="204160"
       TERMINAL_BUFFER_SIZE="1024"
       TYPEAHEAD="Y"
DEFINE GLOBAL_DEFAULT
       8_BIT_ENCODING="Y"
       GLOBAL_JOURNAL="N"
       GLOBAL_COLLATING="NUMERIC"
       DEFINE PROTECTION
              %_GLOBAL_SYSTEM="RWD"
              %_GLOBAL_WORLD="R"
              %_GLOBAL_GROUP="R"
              %_GLOBAL_UCI="RWD"
              GLOBAL_SYSTEM="RWD"
              GLOBAL_WORLD="NONE"
              GLOBAL_GROUP="NONE"
              GLOBAL_UCI="RWD"
DEFINE MAPPED_ROUTINES
       PAGE_CLUSTER_SIZE="8"
       DIRECTORY_NAME="DISK$USER:[VAXDSM]"
       SECTION_TYPE="S"
       UCI_NAME="MGR"
       VOLUME_SET_NAME="EMP"
NON_PAGED_MEMORY="N"
DEFINE HISTOGRAM
       BUFFER_SIZE="50"
```

Example B–1 Cont'd on next page

Example B–1 (Cont.) Sample Configuration Option File

```
DEFINE SCHEMA
       LOCK_UCI_VOL="MGR,EMP"
       READ_UCI_VOL="MGR,EMP"
       MODIFY_UCI_VOL="MGR,TST"
DEFINE TRANSACTION_PROCESSING
       MAXIMUM_TRANSACTIONS="12"
       TP_SINGLE_REFERENCE="Y"
       REDO_BUFFERS="36"
       DEFINE TP_MONITOR
              CHECKPOINT_TIMER="180"
              CLUSTER_TRANSACTIONS="24"
              INTERLOCK_TABLE_SIZE="112"
DEFINE TRANSLATION
       GLOBAL_NAME="EMP"
       OLD_UCI_VOL="MGR,EMP"
       NEW_UCI_VOL="MGR,TST"
       8_BIT_ENCODING="Y"
       GLOBAL_COLLATING="NUMERIC"
DEFINE TRANSLATION
       GLOBAL_NAME="TEST"
       OLD_UCI_VOL="MGR,EMP"
       NEW_UCI_VOL="MGR,TST"
       8_BIT_ENCODING="Y"
       GLOBAL_COLLATING="NUMERIC"
USERS="128"
```

Index

A

Access control
  Application Mode, 3–10, 3–12
  application systems, 3–1
  callable interface, 3–20
  configurations, 3–3, 3–7
  ^CONFIG Utility, 3–7
  DSM$DSM routine, 3–20
  DSM$INITIALIZE routine, 3–20
  environment, 3–2, 3–4
  globals, 3–3, 3–10
  Manager Mode, 3–8
  Programmer Mode, 3–9, 3–12
  relationships, 3–3
  timesharing system, 3–2
  VMS system, 3–2, 3–4
Access control entries
  See ACE
Access control list
  See ACL
Access mode, 3–7
  Application, 3–10, 3–12
  Manager, 3–8
  Programmer, 3–9, 3–12
  specifier, 2–5
Accounts
  application, 2–7, 3–11, 3–16
  captive, 3–4, 3–17
  environment manager, 1–16, 2–6
  programmer, 2–7, 3–11
  tied application, 3–16
  VMS system, 2–1
  VMS users, 3–4
ACE, 3–5
  default protection, 3–6
ACL, 1–17, 2–4
  file protection, 3–5
^ACLCALL program, 3–15
^ACLSET Utility, 3–15, 6–14
^ACLSET Utility (Cont.)
  running in a VAXcluster environment, 3–14
  using ^ACLCALL, 3–15
^ACL Utility, 3–12, 6–14
  creating alternate environment manager, 2–6
  granting Application Mode access, 3–7
  granting Programmer Mode access, 3–7
  using ^ACLCALL, 3–15
ADDUSER.COM
  adding new users to UAF, 3–11
ADD^ACL Utility, 3–13, 6–14
ADD^TTPORT Utility, 3–19
After-image journal file
  placement, 2–4
After-Image Journaling Utilities
  ^DEJRNL, 6–8
  FILES, 6–9
  ^JOURNAL, 6–8
  ^JRNLIS, 6–9
  SETJRN^%GLOMAN, 6–9
ALFMAINT.COM, 3–16
Alternate environment manager, 3–6
  creating, 2–6
  file access, 2–4
  starting a configuration, 2–17
Alternate root
  configuration shutdown, A–9
Alternate root installation, A–1
  directory structure, A–2
  procedure, A–3
  search lists, A–2
  VAXcluster environment, A–9
Application
  access control, 3–1
  accounts, 2–7, 3–11, 3–16
  dedicated system, 3–1
  optimizing, 5–1
Application Mode
  access control, 3–10, 3–12

Application user account, 2–7, 3–11
  dedicated, 3–16
  establishing, 3–11
ASTLM process quota
  requirements for Incremental Backup Utility, 4–2
AUTOGEN.COM
  modifying SYSGEN parameters, 1–7, 4–10
  running, 1–7
Automatic login facility
  See ALFMAINT.COM

B

Backup
  system disk, 1–5
^BACKUP Utility, 6–10
  ASTLM requirements, 4–2
BALSETCNT parameter, 4–4
Baseline mode, 2–5
  access mode specifier, 2–5
^BIJEXAM Utility, 6–9
^BIJFORC Utility, 6–9
^BIJINIT Utility, 6–9
BIJ Utilities
  ^BIJEXAM, 6–9
  ^BIJFORC, 6–9
  ^BIJINIT, 6–9
BYPASS qualifier
  restricting access, 3–9
BYTLM process quota, 4–2

C

Callable interface
  access control, 3–20
Captive accounts
  command procedure, 3–17
  creating, 3–17
  definition of, 3–4
Clusterwide configuration, 2–22
CMKRNL privilege
  installing shared images, 1–15
Commands
  VIEW, 3–9
  ZESCAPE, 3–10
^CONDDP Utility, 6–11
Configuration access global
  See ^DSMACC configuration access global
Configuration identifier, 2–9
Configuration Management Utilities, 6–5
  ^ACL, 6–14
  ^CONFIG, 6–5
  COPY^CONFIG, 6–5
Configuration Management Utilities (Cont.)
  ^DBSMAN, 6–8
  DELETE^CONFIG, 6–5
  DISPLAY^CONFIG, 6–5
  ^FORCEX, 6–15
  ^LOCKTAB, 6–15
  ^LOGIN, 3–21, 6–14
  ^MEMORY, 4–6, 6–14
  MODIFY^CONFIG, 6–5
  ^SHUTDWN, 2–19, 6–15
  ^STU, 2–6, 2–17, 6–14
  ^%SY, 6–14
  ^TIED, 3–18, 6–14
Configuration option file
  converting, B–3
  creating, B–2
  data keywords, B–1, B–4, B–10
  definition keywords, B–1, B–4, B–6
  example, B–21
  minimum entries, B–20
  using, B–1
Configurations
  See also Configuration option file
  access control, 3–3, 3–7
  cluster, 2–22
  database set master, 2–25
  default, 2–19
  defining, 2–9, B–1
  definition of, 2–5
  modifying, 2–20
  option file, B–1
  running multiple, 2–20
  shutting down, 1–27, 2–19
  shutting down in an alternate root, A–9
  starting, 2–17, B–4
  starting in alternate manager account, 2–17
  VAXcluster environment, 2–25
  virtual memory requirements, 4–9
^CONFIG Utility, 2–6, 6–5
  aborting with Ctrl/c, 2–9
  access control, 3–7
  creating a configuration, 2–9 to 2–17
  enabling mapped routines, 5–7
^CONMAN Utility, 2–9, 6–5
COPY^CONFIG Utility, 6–5, 6–7
Ctrl/c
  aborting ^CONFIG Utility, 2–9

D

DASL software
  installation, 1–9
Database

Database (Cont.)
degradation, 1–27
overview, 2–7
Database files
default directory, 2–3
Database set, 2–7
DSM-11 compatible, 2–8
transaction, 2–8
utilities, 6–10
VAX DSM, 2–8
Database set master, 2–25
Data keywords, B–1, B–10
^DBSMAN Utility, 6–8
^DBT Utility, 6–11
DCL commands
DIRECTORY/SECURITY, 3–6
SHOW MEMORY, 4–12
STOP/ID, 2–19
DDP
^CONDDP, 6–11
^DDP, 6–11
^DDPCIR, 6–11
^DDPLNK, 6–11
^DDPSEC, 6–12
^DDPSTA, 6–12
^DDPUTL, 6–11
DECnet
link servers, 2–27
servers, 2–27
START^DDPLNK, 6–12
STOP^DDPLNK, 6–12
^DDPCIR Utility, 6–11
^DDPLNK Utility, 6–11
^DDPSEC Utility, 6–12
^DDPSTA Utility, 6–12, 6–13
^DDP Utility, 6–11
^DDPUTL Utility, 6–11
DECnet
link servers, 2–27
Definition keywords, B–1, B–6
^DEJRNL Utility, 6–8
DELETE^ACL Utility, 3–14, 6–14
DELETE^CONDBS Utility, 6–10
DELETE^CONFIG Utility, 6–5, 6–8
DELETE^TTPORT Utility, 3–20
Detached process, 2–25
DDP servers, 2–27
DECnet link servers, 2–27
Garbage Collector, 2–26
Journal Process, 2–27
Recovery Process, 2–27
Write Demon, 2–26
DIOLM process quota, 4–2
Directories
alternate root, A–2
cluster nodes, 2–3
database files, 2–3
environment root, 2–2
hierarchy, 2–23
journal files, 2–2
single-node system, 2–4
system root, A–1
VAXcluster environment, 2–23
VAX DSM, 2–2
DIRECTORY/SECURITY command (DCL), 3–6
DISPLAY^CONDBS Utility, 6–10
DISPLAY^CONFIG Utility, 6–5
Distributed Data Processing Utilities
See also DDP
^DDP
^CONDDP, 6–11
^DDPCIR, 6–11
^DDPLNK, 6–11
^DDPSEC, 6–12
^DDPSTA, 6–12
^DDPUTL, 6–11
START^DDPLNK, 6–12
STOP^DDPLNK, 6–12
Distribution kit, 1–1
DSM$ALTROOT.COM, A–2
DSM$ALTROOT_INSTALL.COM, A–3
example, A–5
DSM$BATCH batch queue, 2–18
DSM$CONFIGURE.COM
alternate root installation, A–3
configuring VAX DSM, 1–24
DSM$DEFAULT_envnm logical name, 2–21
DSM$DSM routine
access control, 3–20
DSM$ENVIRONMENT logical name, 2–22
DSM$ID_envnm logical name, 2–21
DSM$INIT.COM
creating a VMS account, 3–5
creating directories, 2–3
creating environment manager account, 1–16
directory protection, 2–4
file protection, 2–4
running, 1–18
VAXcluster environment, 2–22
DSM$INITIALIZE routine
access control, 3–20
DSM$INSTALL.COM
allocating nonpaged memory, 4–15

DSM$INSTALL.COM (Cont.)
initializing VAX DSM, 1–26
installing shared images, 1–15
starting a configuration, 2–18
DSM$IVP.COM, 1–9
DSM$MESSAGE.EXE, 1–14
DSM$MT_EBCASC.EXE, 1–15
DSM$SHARE.EXE, 1–14
DSM$SHUTDOWN.COM, 2–19
shutting down a configuration, 1–27
DSM$STARTUP.COM, 2–18
DSM$SYSBUF global section, 4–16
DSM$ZCALL.EXE, 1–15
DSM.EXE, 1–14
DSM-11 compatible database set, 2–8
DSM-11 compatible disk
VMS identifier, 1–17
DSMACC.GBL configuration access global
See ^DSMACC configuration access global
^DSMACC configuration access global, 1–18, 2–2
DSMDEMON.LOG, 2–26
startup failure, 1–25
DSMGARCOL.LOG, 2–27
DSMINIT.COM
command procedure, 1–17
^DSMMGR system global, 1–17

E

ENQLM process quota, 4–2
Environment
See also Environment manager account
access control, 3–2, 3–4
definition of, 2–1
directories, 2–2
identifier, 2–2, 2–20
maintaining security, 3–6
name, 2–21
ENVIRONMENT=envnm qualifier, 2–21
Environment configuration global
See ^SYS environment configuration global
Environment manager
creating user accounts, 3–11
Environment manager account, 2–1, 2–6
alternate, 3–6
creating, 1–16
privileges, 1–24
Environment root directory, 2–2

F

F$GETSYI lexical function, 1–6, 5–4
File protection
ACL protection, 3–5
File protection (Cont.)
identifiers, 3–5
renaming files, 3–7
UIC-based, 2–4, 3–5
FILLM process quota, 4–2
^FORCEX Utility, 2–19, 6–15
Functions
$VIEW, 3–9

G

Garbage Collector
log file, 2–27
process, 2–26
GBLPAGES parameter, 4–5, 5–4
requirements for installation, 1–6
GBLPAGFIL parameter, 4–5
GBLSECTIONS parameter, 4–4, 5–4
requirements for installation, 1–6
Globals
access control, 3–3, 3–10
performance, 5–13
Global sections
shared, 4–12, 5–2
^%GLOMAN Utility, 6–10
^GLSSTA Utility, 5–13, 6–13
report example, 5–14

H

HALT^RTH Utility, 5–20
Help text, 6–2
^%HELP Utility, 6–2
Histograms
database event counts, 5–22
derivative ratios, 5–22
Distributed Data Processing activity, 5–32
global references, 5–30
routine execution, 5–26
routine loads, 5–28

I

Identifiers
file protection, 3–5
VAX DSM environment, 1–17, 2–2, 2–4
Image
installing shared, 1–15
main, 1–14
relationships, 1–14
shareable, 1–14
Incremental backup
ASTLM requirements, 4–2
Installation
alternate root, A–1
backing up system disk, 1–5

Installation (Cont.)
DASL software, 1–9
distribution kit, 1–1
documentation, 1–2
editing system files, 1–26
IVP, 1–9
overview, 1–4
post, 1–16
procedure, 1–5
registering VAX DSM license, 1–5
shared images, 1–15
software, 1–1
space requirements, 1–4
SYSGEN requirements, 1–6
system shutdown, 1–27
system startup, 1–26
time requirements, 1–4
VAXcluster environment, 1–15
VMSINSTAL, 1–7
VMS version required, 1–4
Installation Verification Procedure
See IVP
IVP, 1–9
installing shared images, 1–15

J

Journal Process, 2–27
^JOURNAL Utility, 6–8
^JRNLIS Utility, 6–9

L

Lexical functions
F$GETSYI, 5–4
Library Utilities, 6–1
editing restrictions, 3–9
^%HELP, 6–2
with mapped routines, 5–5
License Management Facility
See LMF
^LKMSTA Utility, 6–13
LMF
registering VAX DSM license, 1–5
VAXcluster environment, 1–15
LOCKIDTBL parameter, 4–4
LOCKIDTBL_MAX parameter, 4–4
^LOCKTAB Utility, 6–15
Log files
Garbage Collector
DSMGARCOL.LOG, 2–27
Write Demon
DSMDEMON.LOG, 2–26
Logical names
DSM$DEFAULT_envnm, 2–21
Logical names (Cont.)
DSM$ENVIRONMENT, 2–22
DSM$ID_envnm, 2–21
^LOGIN Utility, 3–21, 6–14
LRPCOUNT parameter, 4–5

M

Main image, 1–14
Manager Mode
access control, 3–8
MANAGER qualifier, 2–9, 2–17
controlling logins, 3–21
Manager Mode, 3–8
Mapped routines, 5–2
changing routine directories, 5–9
considerations, 5–12
creating, 5–5
disabling, 5–10
enabling with ^CONFIG, 5–7
executing, 5–9
Library Utilities, 5–5
mapping, 5–5
memory requirements, 5–4
modifying SYSGEN parameters, 5–4
utilities, 5–3
Mapped Routine Utilities
^RMAP, 5–2, 5–3, 6–12
MAPSET^RMSHO, 5–3, 6–12
MAP^%RD, 5–3, 6–12
^RMBLD, 5–3, 5–5, 6–12
^RMDEL, 5–3, 6–12
^RMDIS, 5–3, 6–12
^RMINS, 5–3, 6–12
^RMSHO, 5–3, 6–12
MAPSET^RMSHO Utility, 5–3, 5–10, 6–12
MAP^%RD Utility, 5–3, 5–9, 6–12
MAXPROCESSCNT parameter, 4–4
^MEMORY Utility, 4–6, 6–14
^MEMREQ Utility, 1–25, 4–6
private sections, 4–9
process quota requirements, 4–8
requirements for mapped routines, 5–4
resource requirements, 4–8
shared sections, 4–9
SYSGEN parameters, 4–3, 4–7
^MEMUSE Utility, 4–6, 4–16
MODIFY^CONDBS Utility, 2–8, 6–10
MODIFY^CONFIG Utility, 6–5, 6–7
MODIFY^TTROU Utility, 3–19
MODPARAMS.DAT, 1–7, 4–11
MONITOR^GLSSTA Utility, 5–13

N

Nonpaged memory
  configuring, 4–14
  description, 4–13
  electing to use, 4–12

O

Online help
  ^%HELP Utility, 6–2
  using while developing routines, 6–2
OPER privilege, 1–24, 3–12
^OPTBLD Utility, 6–13, B–2
^OPTCON Utility, 6–12, B–3
Option file
  See Configuration option file
Option File Management Utilities
  ^OPTMAN
    ^OPTBLD, 6–13
    ^OPTCON, 6–12
OPTIONS N parameter
  printing release notes, 1–8
^OPTMAN Utility, B–2
OPT^STU Utility
  starting a configuration, B–4

P

Page file, 4–11
  monitoring usage, 4–12
PAK, 1–5
Parameters
  See also SYSGEN parameters
  GBLPAGES, 1–6
  GBLSECTIONS, 1–6
  modifying VMS system, 4–10
  SPTREQ, 4–14
Password
  VMS user accounts, 3–4
Performance Utilities, 6–13
  ^DDPSTA, 6–13
  ^GLSSTA, 5–13, 6–13
  ^LKMSTA, 6–13
  ^RTH, 5–18, 6–13
Physical memory
  determining with the ^MEMORY Utility, 4–6
PHYSICAL_PAGES=nnnn qualifier, 4–7, 4–14
Postinstallation
  creating environment manager's account, 1–16
PQL_DWSDEFAULT parameter, 4–5
PQL_DWSEXTENT parameter, 4–5
PQL_DWSQUOTA parameter, 4–5
PRINT^ACL Utility, 3–15, 6–14
Private sections
  ^MEMREQ Utility, 4–9
Privileges
  application users, 3–16
  CMKRNL, 1–15
  environment manager, 1–24
  for using DCL REPLY/TO command, 1–24
  for using ZUSE command, 1–24
  OPER, 1–24, 3–12
  PRMMBX, 3–12
  programmers, 3–11
  SYSNAM, 3–12
  TMPMBX, 1–24, 3–12
  VMS system, 3–11
PRMMBX privilege, 3–12
Process quotas, 4–1
  ASTLM, 4–2
  BYTLM, 4–2
  determining with the ^MEMORY Utility, 4–6
  determining with the ^MEMREQ Utility, 4–8
  DIOLM, 4–2
  ENQLM, 4–2
  FILLM, 4–2
  WSDEFAULT, 4–2
  WSEXTENT, 4–3
  WSQUOTA, 4–3
Product Authorization Key
  See PAK
Programmer account, 2–7
  establishing, 3–11
  VMS privileges, 3–11
Programmer Mode
  access control, 3–9, 3–12
  access restrictions, 3–9

Q

Qualifiers
  BYPASS, 3–9
  ENVIRONMENT=envnm, 2–21
  MANAGER, 2–9, 2–17, 3–21
  PHYSICAL_PAGES=nnnn, 4–14

R

RECORD^GLSSTA Utility, 5–13
Recovery Process, 2–27
REHASHTBL parameter, 4–4
REPLAY^GLSSTA Utility, 5–13

RIGHTSLIST.DAT, 3–5
^RMAP Utility, 5–2, 5–3, 6–12
^RMBLD Utility, 5–3, 5–5, 6–12
^RMDEL Utility, 5–3, 5–10, 6–12
^RMDIS Utility, 5–3, 6–12
^RMINS Utility, 5–3, 5–8, 6–12
^RMSHO Utility, 5–3, 6–12
^RTHISTP Utility, 5–19
^RTHIST Utility
  histograms
    database event counts, 5–21
    derivative ratios, 5–21, 5–25
    Distributed Data Processing activity, 5–32
    global references per second, 5–30
    routine execution, 5–26
    total routine loads, 5–28
  performance data, 5–20
  queue data, 5–20
^RTH Utility, 5–18, 6–13
  optimizing VAX DSM, 5–34

S

SAVE^RTHISTP Utility, 5–19
SAVE^RTHIST Utility
  creating mapped routines, 5–6
Search lists, A–2
SETJRN^%GLOMAN Utility, 6–9
Shareable image, 1–14
Shared global sections, 4–12
Shared image
  installing VAX DSM, 1–15
  VAXcluster environment, 1–15
Shared memory
  electing to use, 4–12
  nonpaged, 4–13
    configuring, 4–14
  shared global sections, 4–12
Shared sections
  ^MEMREQ Utility, 4–9
SHOW MEMORY command (DCL), 4–12
SHOW^TTPORT Utility, 3–20
SHOW^TTROU Utility, 3–20
^SHUTDWN Utility, 2–19, 6–15
Software Support Addendum
  See SSA
SPTREQ parameter, 4–14
SSA, 1–1
Startup command file, 2–9
START^DDPLNK Utility, 6–12
STATUS^RTH Utility, 5–20
STOP/ID command (DCL), 2–19
STOP^DDPLNK Utility, 6–12
^STU Utility, 2–6, 2–17, 6–14
  OPT^STU Utility, B–4
Swap file, 4–11
  monitoring usage, 4–12
SWAPFILES.COM, 4–12
SYS$EXAMPLES:ADDUSER.COM, 3–11
SYS$MANAGER:ALFMAINT.COM, 3–16
SYS$MANAGER:DSM$CONFIGURE.COM, 1–24
  alternate root installation, A–3
SYS$MANAGER:DSM$INIT.COM, 1–16, 2–22, 3–5
  running, 1–18
SYS$MANAGER:DSM$SHUTDOWN.COM, 1–27, 2–19
SYS$MANAGER:SYSHUTDWN.COM
  shutting down VAX DSM, 1–27
SYS$STARTUP:DSM$INSTALL.COM, 1–15, 1–26, 2–18
SYS$STARTUP:SYSTARTUP_V5.COM
  starting up VAX DSM, 1–26
SYS$SYSTEM:MODPARAMS.DAT, 1–7, 4–3, 4–11
SYS$SYSTEM:RIGHTSLIST.DAT, 3–5
SYS$SYSTEM:SYSUAF.DAT, 3–11
SYS$TEST:DSM$IVP.COM, 1–9
SYS$UPDATE:AUTOGEN.COM, 1–7, 4–10
SYS.GBL global, 2–2
^SYS environment configuration global, 1–17, 2–2
SYSGEN, 4–3, 4–7
  inspecting parameter values, 4–10
  installation requirements, 1–6
SYSGEN parameters
  BALSETCNT, 4–4
  determining with the ^MEMORY Utility, 4–6
  GBLPAGES, 4–5, 5–4
  GBLPAGFIL, 4–5
  GBLSECTIONS, 4–4, 5–4
  LOCKIDTBL, 4–4
  LOCKIDTBL_MAX, 4–4
  LRPCOUNT, 4–5
  MAXPROCESSCNT, 4–4
  ^MEMREQ Utility values, 4–7
  modifying, 4–10
  PQL_DWSDEFAULT, 4–5
  PQL_DWSEXTENT, 4–5
  PQL_DWSQUOTA, 4–5
  REHASHTBL, 4–4
  SPTREQ, 4–14

SYSGEN parameters (Cont.)
tailoring, 4–3
WSMAX, 4–4
SYSHUTDWN.COM, 1–27
SYSMAN Utility (VMS), 1–16
SYSNAM privilege, 3–12
System files
editing, 1–26
System Generation Utility
See SYSGEN
System Management Utilities
^ACL, 3–12, 6–14
^FORCEX, 2–19, 6–15
^LOCKTAB, 6–15
^LOGIN, 6–14
^MEMORY, 6–14
^SHUTDWN, 6–15
^STU, 6–14
^%SY, 6–14
^TIED, 6–14
System shutdown
using DSM$SHUTDOWN.COM, 1–27
System startup
using DSM$INSTALL.COM, 1–26
System Utilities, 6–1
^ACL, 3–12, 6–14
^ACLSET, 3–15, 6–14
ADD^ACL, 3–13, 6–14
ADD^TTPORT, 3–19
After-Image Journaling, 6–8
FILES, 6–9
^BACKUP, 6–10
Before-Image Journaling, 6–9
^BIJEXAM, 6–9
^BIJFORC, 6–9
^BIJINIT, 6–9
^CONDDP, 6–11
^CONFIG, 2–6, 2–9 to 2–17, 3–7, 6–5
Configuration Management, 6–5
^CONMAN, 2–9, 6–5
conventions, 6–2
COPY^CONFIG, 6–5, 6–7
^DBSMAN, 6–8
^DBT, 6–11
^DDP, 6–11
^DDPCIR, 6–11
^DDPLNK, 6–11
^DDPSEC, 6–12
^DDPSTA, 6–12, 6–13
^DDPUTL, 6–11
^DEJRNL, 6–8
DELETE^ACL, 3–14, 6–14
System Utilities (Cont.)
DELETE^CONDBS, 6–10
DELETE^CONFIG, 6–5, 6–8
DELETE^TTPORT, 3–20
DISPLAY^CONDBS, 6–10
DISPLAY^CONFIG, 6–5
Distributed Data Processing, 6–11
^FORCEX, 2–19, 6–15
^%GLOMAN, 6–10
^GLSSTA, 5–13, 6–13
HALT^RTH, 5–20
^JOURNAL, 6–8
^JRNLIS, 6–9
^LKMSTA, 6–13
^LOCKTAB, 6–15
^LOGIN, 3–21, 6–14
Mapped Routines, 5–3, 6–12
MAPSET^RMSHO, 5–3, 5–10, 6–12
MAP^%RD, 5–3, 5–9, 6–12
^MEMORY, 4–6, 6–14
^MEMREQ, 1–25, 4–6
^MEMUSE, 4–6, 4–16
MODIFY^CONDBS, 2–8, 6–10
MODIFY^CONFIG, 6–5, 6–7
MODIFY^TTROU, 3–19
MONITOR^GLSSTA, 5–13
^OPTBLD, 6–13, B–2
^OPTCON, 6–12, B–3
^OPTMAN, 6–12, B–2
OPT^STU, B–4
Performance, 6–13
PRINT^ACL, 3–15, 6–14
RECORD^GLSSTA, 5–13
REPLAY^GLSSTA, 5–13
^RMAP, 5–2, 5–3, 6–12
^RMBLD, 5–3, 5–5, 6–12
^RMDEL, 5–3, 5–10, 6–12
^RMDIS, 5–3, 6–12
^RMINS, 5–3, 5–8, 6–12
^RMSHO, 5–3, 6–12
^RTH, 5–18, 6–13
^RTHISTP, 5–19
running, 6–4
SAVE^RTHISTP, 5–19
SETJRN^%GLOMAN, 6–9
SHOW^TTPORT, 3–20
SHOW^TTROU, 3–20
^SHUTDWN, 2–19, 6–15
START^DDPLNK, 6–12
STATUS^RTH, 5–20
STOP^DDPLNK, 6–12
^STU, 2–6, 2–17, 6–14, B–4
^%SY, 6–14

System Utilities (Cont.)
^SYS, 6–4
^TIED, 3–17, 3–18, 6–14
^TIEDSTU, 3–18
^UCIMAN, 6–10
^UPGCOM, 6–11
^VOLINH, 6–11
^VOLMAN, 2–7, 6–11
^%VOLTAB, 6–10
Volume Set, 6–10
SYSUAF.DAT, 3–11
^SYS Utility, 6–4
^%SY Utility, 6–14

T

Tied application accounts, 3–16
captive accounts, 3–17
captive command procedure, 3–17
establishing, 3–16
Tied routines
port tables, 3–18
Tied Routine Table, 3–19
Tied Routine Table
creating, 3–19
^TIEDSTU Utility, 3–18
Tied terminals, 3–16
captive accounts, 3–17
captive command procedure, 3–17
establishing, 3–16
^TIED Utility, 3–17, 3–18, 6–14
Timesharing system
access control, 3–2
TMPMBX privilege, 1–24, 3–12
Transaction database set, 2–8

U

UAF, 3–11
^UCIMAN Utility, 6–10
UIC
file protection, 2–4, 3–5
^UPGCOM Utility, 6–11
User access
controlling, 3–1
User accounts
controlling access, 3–1
establishing, 3–11
VMS Authorize Utility, 3–11
VMS password, 3–4
User authorization file
See UAF
User identification code
See UIC

V

VAXcluster environment
^ACLSET Utility, 3–14
adding nodes, 2–24
alternate root installation, A–9
defining configurations, 2–25
directory structure, 2–23
enabling DDP, 2–25
initializing VAX DSM in, 2–22
installing VAX DSM in, 1–15
registering license, 1–15
shared image, 1–15
VAX DSM database set, 2–8
VIEW command
restricting access, 3–9
$VIEW function
restricting access, 3–9
Virtual memory
requirements, 4–9
VMS Authorize Utility, 2–7, 3–11
creating captive accounts, 3–17
modifying process quotas, 4–1
VMS free list, 4–14
VMS identifier
See Identifiers
VMSINSTAL
console output, 1–10
installing VAX DSM, 1–7
OPTIONS N, 1–8
VMS Install Utility, 1–6, 1–15, 3–21, 4–16
VMSLICENSE.COM, 1–5
VMS Monitor Utility, 4–5
VMS system
access control, 3–2, 3–4
accounts, 2–1
modifying parameters, 4–10
process quotas, 4–1
tuning for VAX DSM, 4–1
VMS System Management Utility
See SYSMAN Utility (VMS)
VMS users
accounts, 2–6, 3–4
VMS Utilities
Authorize, 2–7, 3–11
automatic login facility, 3–16
Install, 1–6, 1–15, 3–21, 4–16
Monitor, 4–5
SYSGEN, 4–7
SYSMAN, 1–16
VMSINSTAL, 1–7
^VOLINH Utility, 6–11

^VOLMAN Utility, 2–7, 6–11
^%VOLTAB Utility, 6–10
Volume sets
  initializing, 2–7
Volume Set Utilities
  ^BACKUP, 6–10
  Database Repair, 6–10
  ^%GLOMAN, 6–10
  ^UCIMAN, 6–10
  ^UPGCOM, 6–11
  ^VOLINH, 6–11
  ^VOLMAN, 6–11
  ^%VOLTAB, 6–10
^VOLUTL Utility, 6–10

W

Write Demon
  log file, 2–26
  page file considerations, 4–11
  process, 2–26
WSDEFAULT process quota, 4–2
WSEXTENT process quota, 4–3
WSMAX parameter, 4–4
WSQUOTA process quota, 4–3

Z

$ZCALL functions
  %SETUCI
    changing mapped routine directories, 5–9
    restricting access, 3–9
ZESCAPE command
  restricting access, 3–10
ZUSE command
  OPER privilege, 1–24