digital

A Comparison of System Management on OpenVMS AXP and OpenVMS VAX

OpenVMS

Part Number: AA–PV71A–TE

A Comparison of System Management on OpenVMS AXP and OpenVMS VAX

Order Number: AA–PV71A–TE

May 1993

This manual compares system management on OpenVMS AXP and OpenVMS VAX systems. It is intended for experienced system managers who need to learn quickly how specific tasks differ or remain the same on OpenVMS AXP and OpenVMS VAX. The comparison is between OpenVMS AXP Version 1.5 and the following releases: VMS Version 5.4, VMS Version 5.5, and OpenVMS VAX Version 6.0.

Revision/Update Information:	This manual supersedes *A Comparison of System Management on OpenVMS Alpha and OpenVMS VAX,* Version 1.0.
Software Version:	OpenVMS AXP Version 1.5

Digital Equipment Corporation
Maynard, Massachusetts

May 1993

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.

The 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 on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

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

ZK6010

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

3 Maintenance Tasks

4 Security Tasks

5 Performance Optimization Tasks

6 Network Management Tasks

A Overview of OpenVMS AXP Performance Characteristics

B I/O Subsystem Configuration Commands in SYSMAN

C Additional Considerations

Figures

Tables

Preface

Intended Audience

This document is intended for experienced OpenVMS VAX system managers who are establishing an OpenVMS computing environment on AXP computers.

Document Structure

This document consists of six chapters, three appendixes, and an index.

Chapter 1 explains why there are differences in OpenVMS system management on AXP and VAX computers.

Chapter 2 describes how OpenVMS system management setup tasks are similar or different on AXP and VAX computers.

Chapter 3 describes how OpenVMS system management maintenance tasks are similar or different on AXP and VAX computers.

Chapter 4 describes how OpenVMS system management security tasks are similar or different on AXP and VAX computers.

Chapter 5 describes how the OpenVMS system management tasks that are designed to optimize performance are similar or different on AXP and VAX computers. Related information appears in Appendix A.

Chapter 6 describes how DECnet network management tasks are similar or different on AXP and VAX computers.

Appendix A presents performance considerations for users of OpenVMS AXP systems. The information in this appendix supports the performance comparison described in Chapter 5.

Appendix B contains reference descriptions of the I/O subsystem configuration commands that are in the System Management utility (SYSMAN).

Appendix C contains additional system management considerations that might be pertinent to your job of supporting OpenVMS general users and programmers working on AXP computers.

Associated Documents

Depending on your experience level, this manual should provide most of the basic information you will need to get started with the management of OpenVMS AXP systems. However, you should read the following two documents thoroughly:

- *OpenVMS AXP Version 1.5 Release Notes*
- *OpenVMS AXP Version 1.5 Upgrade and Installation Manual*

If your computing environment will include DECwindows Motif for OpenVMS AXP, read:

- *DECwindows Motif for OpenVMS AXP Version 1.1 Release Notes*
- *DECwindows Motif for OpenVMS AXP Version 1.1 Installation Guide*

You might also want to consult the following OpenVMS manuals for detailed system management information:

- *OpenVMS DCL Dictionary*
- *VMScluster Systems for OpenVMS*
- *OpenVMS System Manager's Manual: Essentials*
- *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*
- *OpenVMS System Management Utilities Reference Manual*
- *DECnet for OpenVMS Networking Manual*
- *DECnet for OpenVMS Network Management Utilities*
- *OpenVMS AXP Guide to System Security*
- *OpenVMS VAX Guide to System Security*
- *OpenVMS AXP System Dump Analyzer Utility Manual*
- *OpenVMS VAX System Dump Analyzer Utility Manual*

Conventions

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

All references to DECwindows in this manual refer to DECwindows Motif for OpenVMS AXP software.

The following conventions are also used in this manual:

Ctrl/*x*
A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.

.
.
.
A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.

( )
In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.

[ ]
In format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in an OpenVMS file specification, or in the syntax of a substring specification in an assignment statement.)

{ }
In format descriptions, braces surround a required choice of options; you must choose one of the options listed.

boldface text

Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason.

Boldface text is also used to show user input in Bookreader versions of the manual.

italic text

Italic text emphasizes important information, indicates variables, and indicates complete titles of manuals. Italic text also represents information that can vary in system messages (for example, Internal error *number*), command lines (for example, /PRODUCER=*name*), and command parameters in text.

UPPERCASE TEXT

Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.

-

A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.

numbers

All numbers in text are assumed to be decimal, unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1 Overview of OpenVMS AXP System Management

Virtually all OpenVMS VAX system management utilities, command formats, and tasks are identical in the OpenVMS AXP environment. However, some important exceptions exist that experienced OpenVMS VAX system managers must consider to properly set up, maintain, secure, optimize, and establish network connections for OpenVMS AXP systems.

Read this manual if you know about most OpenVMS VAX system management features and only need to learn what is new, identical, or different in OpenVMS AXP system management.

This manual compares system management of OpenVMS AXP Version 1.5 with:

- VMS Version 5.4
- VMS Version 5.5
- OpenVMS VAX Version 6.0

A goal of this manual is to ease your system management migration as much as possible.

This chapter explains why some differences exist between OpenVMS AXP and OpenVMS VAX system management. In subsequent chapters:

- Chapter 2 compares the *setup* features and tasks.
- Chapter 3 compares the *maintenance* features and tasks.
- Chapter 4 compares the *security* features and tasks.
- Chapter 5 compares the *performance optimization* features and tasks.
- Chapter 6 compares the *network* management features and tasks.

You will find supporting system management information in the appendixes to this document. To understand why OpenVMS tuning is different on AXP computers, read Appendix A. Appendix B contains reference material for the I/O subsystem configuration capabilities that have moved from the System Generation utility (SYSGEN) to the System Management utility (SYSMAN). Appendix C describes additional considerations related to your task of supporting general users and programmers on OpenVMS AXP systems.

1.1 Why Does System Management Differ on AXP and VAX?

If "VMS is VMS" on AXP and VAX, then why are there *any* differences in system management? The following sections summarize the significant reasons for the differences.

The remaining chapters in this document will help you identify the OpenVMS AXP and OpenVMS VAX system management characteristics.

1.1.1 Different Page Size

OpenVMS VAX and OpenVMS AXP systems allocate and deallocate memory for processes in units called **pages**. A page on a VAX system is 512 bytes. On AXP systems, the page size will be one of four values, 8 kilobytes (KB) (8192 bytes), 16KB, 32KB, or 64KB. A particular AXP system will implement only one of the four page sizes and the initial set of AXP computers use an 8KB page.

This difference in page size is significant to OpenVMS system managers in two ways:

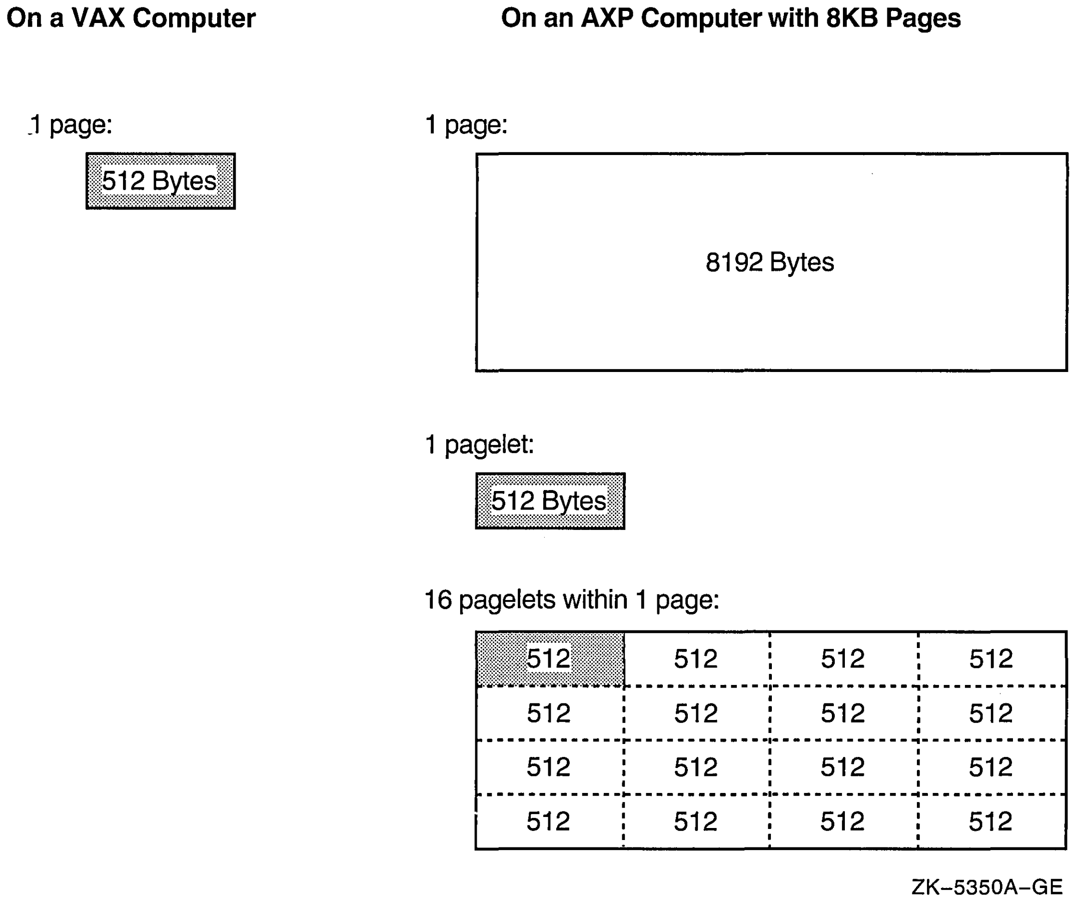
- You might need to adjust process quotas and limits, and system parameters, to account for the additional resources (especially memory resources) users might require. For example, higher values for the PGFLQUOTA process quota and the GBLPAGES system parameter might be necessary.
- In a number of cases, OpenVMS AXP interactive utilities present to and accept from users units of memory in a 512-byte quantity called a **pagelet**. Thus, one AXP pagelet is the same size as one VAX page. Also, on an AXP computer with 8KB pages, 16 AXP pagelets equal 1 AXP page.

  Internally, for the purposes of memory allocation, deletion, and protection, OpenVMS AXP will round up (if necessary) the value you supply in pagelets to a number of CPU-specific pages.

  The use of pagelets provides compatibility with OpenVMS VAX users, system managers, and application programmers who are accustomed to thinking about memory values in 512-byte units. In a dual-architecture VMScluster, which can include VMS Version 5.5–2 nodes and OpenVMS AXP Version 1.5 nodes, it is helpful to know that a VAX page and an AXP pagelet represent a common unit of 512 bytes. Also, existing OpenVMS VAX applications do not need to change parameters to the memory management system services when the applications are ported to OpenVMS AXP.

Figure 1–1 illustrates the relative sizes of a VAX page, an AXP 8KB page, and an AXP pagelet.

Figure 1–1 VAX Page Size, AXP Page Size, and AXP Pagelet Size



OpenVMS AXP does not allocate or deallocate a portion of a page. The user-interface quantity called a pagelet is not used internally by the operating system. Pagelets are accepted and displayed by utilities so that users and applications operate with the understanding that each VAX page value and each AXP pagelet value equal a common 512-byte quantity.

In your OpenVMS AXP environment, you will need to notice when page or pagelet values are being shown in memory displays. If a memory value represents a page on an AXP, the documentation might refer to "CPU-specific pages." This convention indicates possible significant differences in the size of the memory being represented by the page unit, depending on the AXP computer in use (8KB pages, 16KB pages, 32KB pages, or 64KB pages). In general, OpenVMS AXP utilities display CPU-specific page values when the data represents physical memory.

Page and pagelet units are discussed in many sections of this manual; see especially Section 2.2.25, Section 5.1, Section 5.2, and Section 5.3.

1.1.2 VMSclusters Supported with Configuration Limitations

A **VMScluster** system, known as a VAXcluster system in VAX environments, can consist of either of the following:

- All OpenVMS AXP Version 1.5 nodes
- A combination of OpenVMS AXP Version 1.5 nodes and VMS Version 5.5–2 nodes

A VMScluster environment that includes AXP and VAX nodes is called a dual-architecture VMScluster. Basic file sharing and many other VMScluster features are supported. At this time, VMScluster configurations are limited, as summarized in Section 2.2.1 and as detailed in the *OpenVMS AXP Version 1.5 Release Notes* and in *VMScluster Systems for OpenVMS*.

1.1.3 Unsupported Components and Related Products

Besides the limits to VMScluster configurations, a number of OpenVMS components and related products are not supported on OpenVMS AXP, including:

- Volume Shadowing for OpenVMS.
- OpenVMS RMS Journaling.
- DECnet/OSI for OpenVMS.
- User-written device drivers.
- Distributed Name Service (DNS).
- VAX P.S.I. (However, an OpenVMS AXP node can connect to X.25 and X.29 networks via an X.25 or X.29 router on the same local area network.)
- Level 2 host-based DECnet routing; level 1 routing is supported only for use with cluster alias routers, and also is restricted for use on one circuit.
- DDCMP network connections.

1.1.4 Release Features

In some cases, the differences that you will notice in OpenVMS AXP system management depend on the VAX VMS or OpenVMS VAX release that you are currently using. Some brief examples:

- OpenVMS AXP Version 1.5 includes the enhanced clusterwide batch and print queue failover capabilities that are part of VMS Version 5.5–2. If you are currently using a version of VMS earlier than Version 5.5, the new batch and print features might be new to you. On the other hand, additional batch and print queuing features were added to OpenVMS VAX Version 6.0 and are not present in OpenVMS AXP Version 1.5. See Section 3.2.1 for details.
- OpenVMS AXP Version 1.5 does not include the C2 security features of OpenVMS VAX Version 6.0. A number of system resources, rights database characteristics, and DCL commands are affected by C2 features in OpenVMS VAX Version 6.0. If you are migrating to OpenVMS AXP Version 1.5 from a version of OpenVMS VAX earlier than Version 6.0, then you will not notice any security-related changes. If you are already accustomed to the Version 6.0 C2 features, however, you will notice the pre–Version 6.0 security characteristics on your OpenVMS AXP Version 1.5 systems. See Chapter 4 for details.

- OpenVMS AXP Version 1.5 supports multiadapter Ethernet cluster protocols to provide for networking redundancy and greater application availability. This feature was first released in VMS Version 5.4–3. If your computing environment is migrating to OpenVMS AXP directly from VMS Version 5.4–2, the multiadapter Ethernet cluster protocols might be new to you. See *VMScluster Systems for OpenVMS* for details.

1.1.5 I/O Subsystem Configuration Commands in SYSMAN

On OpenVMS VAX computers, the System Generation utility (SYSGEN) is used to modify system parameters, load device drivers, load page and swap files, and create additional page and swap files. To load device drivers on OpenVMS AXP computers, you use the System Management utility (SYSMAN) instead of SYSGEN. OpenVMS AXP SYSGEN is available for modifying system parameters,[1] loading page and swap files, and creating additional page and swap files. OpenVMS VAX procedures that use commands such as SYSGEN AUTOCONFIGURE ALL must be modified if they are copied to OpenVMS AXP systems as part of your migration effort. See Chapter 2 and Appendix B for details about SYSMAN IO commands.

1.1.6 MONITOR POOL Command Not Provided

The DCL command MONITOR POOL that is used on VMS Version 5.5 and earlier systems is not provided on OpenVMS AXP systems or on OpenVMS VAX Version 6.0 systems. MONITOR POOL functions are replaced by enhanced, adaptive pool management features and two System Dump Analyzer (SDA) commands in OpenVMS AXP. See Section 5.4 for details about adaptive pool management.

1.1.7 Higher Disk Quotas

You might need to increase disk quotas on OpenVMS AXP computers that store translated images from OpenVMS VAX computers and native OpenVMS AXP images. See Section 2.2.27.

1.1.8 Changes in OpenVMS File Names

The names of some command procedure files supplied by the operating system have changed. For example, SYSTARTUP_V5.COM from VMS Version 5.5 and earlier is called SYSTARTUP_VMS.COM on OpenVMS AXP. Also, the VAXVMSSYS.PAR system parameter file is called ALPHAVMSSYS.PAR on OpenVMS AXP. See Chapter 2.

1.1.9 Some Layered Products Not Supported

With OpenVMS AXP Version 1.5, a number of layered products from Digital Equipment Corporation and other vendors are not supported yet. Your Digital representative can provide you with a current list of the Digital layered products that are available for OpenVMS AXP computers.

If you copy existing startup procedures from one of your OpenVMS VAX computers to an OpenVMS AXP system disk, you must comment out the calls to the startup procedures of currently unsupported layered products.

[1] Although SYSGEN is available for modifying system parameters, Digital recommends that you use AUTOGEN and its data files instead or that you use SYSMAN (between boots, for dynamic parameters).

2 System Setup Tasks

System management setup tasks are those you perform to get the OpenVMS system installed, booted, and ready for users. This chapter:

- Identifies which OpenVMS system management setup tasks are the same on AXP and VAX computers
- Explains which OpenVMS system management setup tasks are different or new on AXP computers

2.1 Setup Tasks That Are the Same

Table 2–1 lists the OpenVMS system management setup tasks that are identical or similar on AXP and VAX.

Table 2–1 Identical or Similar VMS Setup Tasks

Feature or Task	Comments
System disk directory structure	Directory structure is the same (SYS$SYSTEM, SYS$MANAGER, SYS$UPDATE, and so on.)
Site-independent STARTUP.COM command procedure	Each OpenVMS AXP release ships a new SYS$SYSTEM:STARTUP.COM procedure. Do not modify STARTUP.COM.
Site-specific startup command procedures	OpenVMS AXP includes the following site-specific startup command procedures: SYPAGSWPFILES.COM, SYCONFIG.COM, SYLOGICAL.COM, and SYSECURITY.COM. The VMS SYSTARTUP_V5.COM procedure is called SYSTARTUP_VMS.COM in OpenVMS AXP and in OpenVMS VAX Version 6.0.
Installing operating system software	Similar process.
Decompressing libraries as a postinstallation task	Use @SYS$UPDATE:LIBDECOMP.COM as usual if you choose to decompress the system libraries (recommended).
VAXclusters	VMSclusters are supported with configuration limitations for OpenVMS AXP Version 1.5. See Section 2.2.1 for more information.

(continued on next page)

Table 2–1 (Cont.) Identical or Similar VMS Setup Tasks

Feature or Task	Comments
Backing up data	With one exception (/EXACT_ORDER qualifier added in OpenVMS VAX Version 6.0), the BACKUP command and qualifiers are the same on OpenVMS AXP. Note: Thoroughly read the *OpenVMS AXP Version 1.5 Release Notes* for the latest information about any restrictions with backup and restore operations on AXP and VAX systems. (See Section 3.2.2 for information about BACKUP /EXACT_ORDER.)
LAT startup	You must start DECnet before you start LAT. As on OpenVMS VAX, always start LAT from the SYSTEM account. This account typically has appropriate privileges and quotas. LAT functions better when the LATACP process is running under UIC [1,4]. You can add the following command to the SYSTARTUP_VMS.COM procedure that resides in the SYS$MANAGER directory: `$ @SYS$STARTUP:LAT$STARTUP.COM`
Local Area Transport Control Program (LATCP)	Features are identical. To use LATCP, set the system parameter MAXBUF to 8192 or higher. Different systems might require different settings for MAXBUF. The BYTLM quota for accounts that use LATCP should be set accordingly in the Authorize utility.
LATSYM symbiont	Identical. Use LATSYM to set up LAT print queues. See the *OpenVMS System Manager's Manual* for more information.
LTPAD process	LTPAD provides outgoing SET HOST/LAT functionality. Service responder and outgoing connections on an AXP computer can be enabled.
DEC InfoServer, which includes the Local Area Disk Control Program (LADCP), LASTport network transport control program, LASTport/Disk protocol, LASTport/Tape protocol	Identical.
ISO 9660 standard	Supported on OpenVMS AXP with restrictions. See Section 2.2.17 for more information.
DECdtm and its two-phase commit protocol	Identical.
VMSTAILOR and DECW$TAILOR utilities	Identical.
Authorize utility	SYSUAF commands and parameters are the same. However, the default values for a number of process quota parameters are higher. See Section 2.2.25 for more information.

(continued on next page)

Table 2–1 (Cont.) Identical or Similar VMS Setup Tasks

Feature or Task	Comments
OPCOM	Identical.
Terminal Fallback Facility (TFF)	Similar function but with some differences. See Section 2.2.28 for details.
User Environment Test Package (UETP)	Identical.

2.2 Setup Tasks That Are Different

This section describes the OpenVMS system management setup tasks that are different or new on AXP. Differences are in the following areas:

- VMSclusters are supported with configuration limitations at this time. See Section 2.2.1.
- Volume Shadowing for OpenVMS is not supported at this time. See Section 2.2.2.
- RMS Journaling is not supported at this time. See Section 2.2.3.
- Planning for and managing common object and image file extensions. See Section 2.2.4.
- The format of the BOOT command on AXP systems and the boot flags are different. See Section 2.2.5.
- The CONSCOPY.COM command procedure is not supplied with the OpenVMS AXP kit. See Section 2.2.6.
- Use of the License Management Facility (LMF). See Section 2.2.7.
- One of the two Product Authorization Keys (PAKs) for DECnet for OpenVMS AXP has a different name from one of the PAK names on VAX systems. See Section 2.2.8.
- DECwindows Motif for OpenVMS AXP (DECwindows) setup tasks. See Section 2.2.9.
- Multihead configuration on a DEC 3000 series computer. See Section 2.2.10.
- System Generation utility (SYSGEN) and its parameters. See Section 2.2.11.
- I/O subsystem configuration commands, controlled in SYSGEN on OpenVMS VAX, are in the OpenVMS AXP System Management utility (SYSMAN). See Section 2.2.12.
- Symmetric multiprocessing (SMP) is supported on OpenVMS AXP Version 1.5. See Section 2.2.13.
- Startup command procedure changes because of relocation of I/O subsystem configuration functions from SYSGEN to SYSMAN. See Section 2.2.14.
- Hardware devices on AXP computers. See Section 2.2.15.
- Digital Storage Architecture (DSA) device naming. See Section 2.2.16.
- The ISO 9660 standard is supported on OpenVMS AXP, but with a number of restrictions at this time. See Section 2.2.17.
- The file-name format of drivers supplied by Digital on AXP systems. Also, user-written device drivers are not supported. See Section 2.2.18.

- OpenVMS installation media contains binaries and documentation. See Section 2.2.19.
- VMSINSTAL utility. See Section 2.2.20.
- Running the AUTOGEN procedure. See Section 2.2.21.
- Improving the performance of main images and shareable images by using a feature called granularity hint regions. See Section 2.2.22.
- SYS.EXE loadable executive image renamed to SYS$BASE_IMAGE.EXE. See Section 2.2.23.
- The security audit log file name is different on OpenVMS VAX Version 6.0. See Section 2.2.24.
- SYSUAF.DAT file and new defaults for process limits and quotas. See Section 2.2.25.
- Special considerations for a common SYSUAF.DAT file in dual-architecture VMScluster environments. See Section 2.2.26.
- Reserving disk space for translated code; also, the possible need for larger page file quota. See Section 2.2.27.
- Terminal Fallback Facility (TFF). See Section 2.2.28.

2.2.1 VMScluster Support

OpenVMS AXP contains a significant number of VMScluster capabilities, including the ability to support VMScluster configurations consisting of only OpenVMS AXP processors, or a dual-architecture configuration combining AXP and VAX processors.

2.2.1.1 VMScluster Features

The VMScluster features for OpenVMS AXP Version 1.5 are:

- Dual-host Digital Storage Systems Interconnect (DSSI) for VMScluster communications
- CI computer interconnect for VMScluster communications
- Ethernet for VMScluster communications
- Booting OpenVMS AXP satellite nodes on OpenVMS AXP platforms
- Booting from multiple Ethernet adapters and multiple adapter run-time support
- Clusterwide shared RMS files
- Clusterwide MOUNT/DISMOUNT
- Clusterwide $BRKTHRU services
- Clusterwide OPCOM, REQUEST, and REPLY
- Clusterwide locking
- MSCP served disks from OpenVMS VAX processors to OpenVMS AXP processors
- MSCP served disks from OpenVMS AXP processors to OpenVMS VAX processors
- Booting OpenVMS AXP nodes from shared DSSI or CI disks

- Access to CI storage subsystems
- Clusterwide SHOW USERS and SHOW SYSTEM
- Clusterwide process services
- Show Cluster utility
- Shared Files–11 On-Disk Structure Level 2 (ODS-2) disks
- Shared mail profile database
- Shared mail files
- Shared SYSUAF files (see Section 2.2.26 for more information)
- Shared RIGHTSLIST
- Shared DECnet database files (except for restrictions on the NETNODE_REMOTE.DAT file; see Section 2.2.1.2 for information about the restriction with OpenVMS AXP satellite booting.)
- Quorum disk support
- Volume sets

2.2.1.2 Configuration Restrictions

The following restrictions apply to VMScluster support in OpenVMS AXP Version 1.5:

- Hardware support. Refer to the VMScluster Software for OpenVMS AXP *Software Product Description* (SPD 42.18.*xx*) for the most up-to-date information about the hardware devices supported with VMSclusters at this time. Your Digital representative can provide you with a copy of the SPD.
- OpenVMS AXP systems cannot access disks that are shadowed using either VAX Volume Shadowing (Phase I) or Volume Shadowing for OpenVMS (Phase II).
- Stripe sets are not supported.
- RMS Journaling is not supported.
- Satellite booting of AXP processors by VAX processors or of VAX processors by AXP processors is not supported.
- Special file naming that distinguishes OpenVMS AXP files from VAX VMS files of the same type (such as for executable images for different architectures) is not provided.
- If satellite booting (VAX VMS or OpenVMS AXP) is being used anywhere in the VMScluster, then the NETNODE_REMOTE.DAT file must not be shared between the VAX and AXP processors. This restriction ensures that OpenVMS AXP systems do not try to downline load VAX systems, and that VAX systems do not try to downline load OpenVMS AXP systems.
- In a VMScluster consisting only of AXP systems, the DECnet cluster alias is supported when at least one of the AXP systems has an extended function license (DVNETEXT) and has level 1 routing enabled. In a dual-architecture VMScluster, at least one of the VAX VMS Version 5.5–2 systems must have a DECnet routing license (DVNETRTG) *or* one of the OpenVMS AXP systems must have a a DECnet extended function license (DVNETEXT). Level 1 routing should be enabled on the nodes with these licenses.

Note how the Product Authorization Key (PAK) name enabling DECnet for OpenVMS AXP cluster alias routing support (DVNETEXT) is different from the PAK name enabling cluster alias routing support on OpenVMS VAX (DVNETRTG). The functions supported with the DVNETEXT license differ from those supplied with the DVNETRTG license. DVNETEXT is provided only to enable cluster alias routing support.

Level 1 DECnet routing is available, but is supported only on DECnet for OpenVMS AXP nodes acting as routers for for a cluster alias. Routing between multiple circuits is not supported. Level 2 routing is not supported on DECnet for OpenVMS AXP nodes.

2.2.1.3 Required VAX VMS Version

VAX nodes running in the VMScluster must be running VAX VMS Version 5.5–2.

2.2.1.4 For Additional Information About VMSclusters

Refer to the *OpenVMS AXP Version 1.5 Release Notes* and *VMScluster Systems for OpenVMS* for details on the following VMScluster topics:

- Preparing the OpenVMS AXP system disk
- Booting OpenVMS AXP CI nodes
- Booting OpenVMS AXP DSSI nodes
- Assigning DSSI bus identifications
- Configuring DSSI device parameters
- Setting host to DSSI disks
- Booting DSSI devices
- OpenVMS AXP satellite booting
- Configuring VMScluster nodes to downline load satellites
- Troubleshooting satellite boot problems

2.2.2 Volume Shadowing for OpenVMS Not Supported

Host-based OpenVMS volume shadowing and controller-based volume shadowing are available on OpenVMS VAX only. Volume shadowing is not available on OpenVMS AXP at this time. Therefore, the MOUNT/SHADOW command will not work on an OpenVMS AXP system.

2.2.3 RMS Journaling Not Supported

RMS Journaling is not supported on OpenVMS AXP at this time. The following commands will not work on an OpenVMS AXP node:

- SET FILE/AI_JOURNAL
- SET FILE/BI_JOURNAL
- SET FILE/RU_ACTIVE
- SET FILE/RU_FACILITY
- SET FILE/RU_JOURNAL
- RECOVER/RMS_FILE

If entered by users, all SET FILE options available for RMS Journaling will return the following error message:

```
%SET-W-JNLNOTAUTH, VAX RMS Journaling not authorized;
operation still performed
-LICENSE-F-NOLICENSE, no license is active for this software product
```

The error message in the previous display is the same error message that would appear on a VAX computer if the RMS Journaling license were not loaded.

If entered by users, the RECOVER/RMS_FILE command will return the following error message:

```
%DCL-W-ACTIMAGE, error activating image RECOVER
-CLI-E-IMAGEFNF, image file not found
$5$DKC200:[SYS0.SYSCOMMON.][SYSEXE]RECOVER.EXE;
```

2.2.4 Planning for and Managing Common Object and Image File Extensions

File extensions on OpenVMS VAX computers are identical on OpenVMS AXP computers, including .OBJ for object files and .EXE for executable files. It is important that you plan for and track the location of the following files, especially in dual-architecture VMScluster systems with common disks:

- Native, VAX specific .OBJ and .EXE files (to be linked or executed on OpenVMS VAX nodes only).
- Native, AXP specific .OBJ and .EXE files (to be linked or executed on OpenVMS AXP nodes only).
- Translated VAX .EXE images (to be executed on OpenVMS AXP nodes only). An OpenVMS VAX image named *file*.EXE becomes *file*_TV.EXE when translated.

The display created by ANALYZE/OBJECT and ANALYZE/IMAGE commands on an OpenVMS AXP node identifies the architecture type of an .OBJ or .EXE file. The OpenVMS AXP command ANALYZE/OBJECT works with AXP or VAX object files. Similarly, the OpenVMS AXP command ANALYZE/IMAGE works with AXP or VAX image files. The OpenVMS VAX ANALYZE/OBJECT and ANALYZE/IMAGE commands do not have this capability.

- When you enter an ANALYZE/IMAGE command on an OpenVMS AXP node and the image being analyzed is an OpenVMS VAX image file, the following text is included on the first page of the displayed report:

  ```
  This is an OpenVMS VAX image file
  ```

- When you enter an ANALYZE/OBJECT command on an OpenVMS AXP node and the object being analyzed is an OpenVMS VAX object file, the following text is included on the first page of the displayed report:

  ```
  This is an OpenVMS VAX object file
  ```

- When you enter an ANALYZE/IMAGE command on an OpenVMS AXP node and the image being analyzed is an OpenVMS AXP image file, the following text is included on the first page of the displayed report:

  ```
  This is an OpenVMS Alpha image file
  ```

- When you enter an ANALYZE/OBJECT command on an OpenVMS AXP node and the object being analyzed is an OpenVMS AXP object file, the following text is included on the first page of the displayed report:

  ```
  This is an OpenVMS Alpha object file
  ```

On an OpenVMS VAX node, the LINK and RUN commands return error messages if the file that users are attempting to link or run was created by an OpenVMS AXP compiler or linker. For example:

```
$ ! On an OpenVMS VAX node
$ RUN SALARY_REPORT.EXE     ! An OpenVMS AXP image

%DCL-W-ACTIMAGE, error activating image SALARY_REPORT.EXE
-CLI-E-IMGNAME, image file _$11$DUA20:[SMITH.WORK]SALARY_REPORT.EXE;1
-IMGACT-F-IMG_SIZ, image header descriptor length is invalid
```

An error message is displayed when you attempt to execute a VAX image on an OpenVMS AXP node. For example:

```
$ ! On an OpenVMS AXP node
$ RUN PAYROLL.EXE           ! An OpenVMS VAX image

%DCL-W-ACTIMAGE, error activating image PAYROLL
-CLI-E-IMGNAME, image file DUA6:[SMITH.APPL]PAYROLL.EXE;7
-IMGACT-F-NOTNATIVE, image is not an OpenVMS Alpha image
```

2.2.5 BOOT Console Command

The AXP console software attempts to locate, load, and transfer the primary bootstrap program from the boot devices specified in the BOOT console command. The BOOT command format on AXP systems is:

BOOT [[-FLAGS system_root,boot_flags] [device_list]]

The –FLAGS qualifier indicates that the next two comma-separated strings are the *system_root* and *boot_flags* parameters. Console software passes both parameters to Alpha primary bootstrap (APB) without interpretation as an ASCII string like 0,0 in the environment variable BOOTED_OSFLAGS.

The *system_root* parameter specifies the hexadecimal number of the root directory on the system disk device in which the bootstrap files and bootstrap programs reside. A root directory is a top-level directory whose name is in the form SYS*nn*, where *nn* is the number specified by *system_root*. (It is recorded by APB in the high-order word of SWRPB$IQ_BOOT_FLAGS.)

The *boot_flags* parameter specifies the hexadecimal representation of the sum of the desired boot flags. Table 2–2 lists possible boot flags and their values.

The *device_list* parameter is a list of device names, delimited by commas, from which the console must attempt to boot. A device name in *device_list* does not necessarily correspond to the OpenVMS device name for a given device. In fact, console software translates the device name to a path name before it attempts the bootstrap. The path name enables the console to locate the boot device through intervening adapters, buses, and widgets (for example, a controller). The path name specification and the algorithm that translates the device name to a path name are system specific.

Table 2–2 Boot Flags and Their Values

Hexadecimal Value	Name	Meaning if Set
1	CONV	Bootstrap conversationally; that is, allow the console operator to modify system parameters in SYSBOOT.
2	DEBUG	Map XDELTA to running system.
4	INIBPT	Stop at initial system breakpoint.
8	DIAG	Perform diagnostic bootstrap.
10	BOOBPT	Stop at bootstrap breakpoints.
20	NOHEADER	Secondary bootstrap image contains no header.
40	NOTEST	Inhibit memory test.
80	SOLICIT	Prompt for the name of the secondary bootstrap file.
100	HALT	Halt before secondary bootstrap.
2000	CRDFAIL	Mark corrected read data error pages bad.
10000	DBG_INIT	Enable verbose mode in APB, SYSBOOT, and EXEC_INIT.
20000	USER_MSGS	Enable descriptive mode, presenting a subset of the verbose mode seen when DBG_INIT is enabled. See Section 2.2.5.1 for more information.

In response to the BOOT console command, console software attempts to boot from devices in the boot device list, starting with the first one. As it attempts to boot from a specific device, console software initializes the BOOTED_DEV environment variable with the path name of that device. If an attempt to boot from a specific device fails, console software attempts to boot from the next device in the list. If all attempts fail, console software prints an error message on the console and enters the halt state to await operator action.

Later, APB uses the value in BOOTED_DEV to determine the boot device.

2.2.5.1 DBG_INIT and USER_MSGS Boot Flags

When the DBG_INIT boot flag (bit 16) is set, many informational messages are displayed during booting. This bit normally is used during testing but could be useful for any problems with booting the computer. Bits <63:48> contain the SYS*n* root from which you are booting.

OpenVMS AXP includes a new flag, USER_MSGS, that enables descriptive booting. This flag is bit 17. Set the USER_MSGS boot flag the same way you set other boot flags.

When the USER_MSGS flag is set, messages that describe the different phases of booting are displayed. These messages guide the user through the major booting phases and are a subset of the messages displayed in verbose mode when the bit 16 DBG_INIT flag is set. The USER_MSGS flag suppresses all the test and debug messages that are displayed when bit 16 is set. Error messages are always enabled and displayed as needed.

The following display shows a partial boot session with the USER_MSGS flag set:

```
INIT-S-CPU...
AUDIT_CHECKSUM_GOOD
AUDIT_LOAD_BEGINS
AUDIT_LOAD_DONE
%APB-I-APBVER, Alpha AXP Primary Bootstrap, Version X59S
%APB-I-BOOTDEV, Determining boot device type
%APB-I-BOOTDRIV, Selecting boot driver
%APB-I-BOOTFILE, Selecting boot file
%APB-I-BOOTVOL, Mounting boot volume
%APB-I-OPBOOTFILE, Opening boot file
%APB-I-LOADFILE, Loading [SYS0.SYSCOMMON.SYSEXE]SYSBOOT.EXE;1
%APB-I-SECBOOT, Transferring to secondary bootstrap
```

In comparison, the following display shows a partial boot session with the DBG_INIT flag set. Notice that many more messages are displayed.

```
INIT-S-CPU...
AUDIT_CHECKSUM_GOOD
AUDIT_LOAD_BEGINS
AUDIT_LOAD_DONE
%APB-I-APBVER, Alpha AXP Primary Bootstrap, Version X59S
Initializing TIMEDWAIT constants...
Initializing XDELTA...
Initial breakpoint not taken...
%APB-I-BOOTDEV, Determining boot device type
Initializing the system root specification...
%APB-I-BOOTDRIV, Selecting boot driver
%APB-I-BOOTFILE, Selecting boot file
%APB-I-BOOTVOL, Mounting boot volume
Boot QIO: VA = 20084000 LEN = 00000024 LBN = 00000000 FUNC = 00000032
Boot QIO: VA = 00000000 LEN = 00000000 LBN = 00000000 FUNC = 00000008
Boot QIO: VA = 20084000 LEN = 00000012 LBN = 00000000 FUNC = 00000027
Boot QIO: VA = 20084000 LEN = 00000008 LBN = 00000000 FUNC = 00000029
Boot QIO: VA = 20086000 LEN = 00000200 LBN = 00000001 FUNC = 0000000C
Boot QIO: VA = 20086200 LEN = 00000200 LBN = 000EE962 FUNC = 0000000C
Boot QIO: VA = 2005DD38 LEN = 00000200 LBN = 000EE965 FUNC = 0000000C
Boot QIO: VA = 20088000 LEN = 00001200 LBN = 00000006 FUNC = 0000000C
%APB-I-OPBOOTFILE, Opening boot file
Boot QIO: VA = 20098000 LEN = 00000200 LBN = 000EEBFE FUNC = 0000000C
Boot QIO: VA = 20089200 LEN = 00000200 LBN = 0000001B FUNC = 0000000C
Boot QIO: VA = 20098000 LEN = 00000200 LBN = 000EEC08 FUNC = 0000000C
Boot QIO: VA = 20089400 LEN = 00000200 LBN = 0013307D FUNC = 0000000C
Boot QIO: VA = 20098000 LEN = 00000200 LBN = 000EE96B FUNC = 0000000C
Boot QIO: VA = 20089600 LEN = 00000200 LBN = 00000027 FUNC = 0000000C
Boot QIO: VA = 20098000 LEN = 00000200 LBN = 000EE975 FUNC = 0000000C
Boot QIO: VA = 20089800 LEN = 00001600 LBN = 000F2B6E FUNC = 0000000C
Boot QIO: VA = 20098000 LEN = 00000200 LBN = 000EE9DB FUNC = 0000000C
%APB-I-LOADFILE, Loading [SYS0.SYSCOMMON.SYSEXE]SYSBOOT.EXE;1
Boot QIO: VA = 2009A000 LEN = 00000200 LBN = 00111993 FUNC = 0000000C
Boot QIO: VA = 00000000 LEN = 00050200 LBN = 00111995 FUNC = 0000000C
%APB-I-SECBOOT, Transferring to secondary bootstrap
```

2.2.6 CONSCOPY.COM Procedure Not Available

The OpenVMS VAX kit provides the CONSCOPY.COM command procedure, which you can use to create a backup copy of the original console volume. The OpenVMS VAX installation supplies the procedure in SYS$UPDATE. The CONSCOPY.COM procedure does not exist for OpenVMS AXP computers as the AXP consoles exist in read-only memory and not on disks.

2.2.7 Use of the License Management Facility (LMF)

Availability Product Authorization Keys (PAKs) are available for OpenVMS AXP. An OpenVMS AXP PAK can be identified by the keyword ALPHA in the PAK's option field.

PAKs having the ALPHA option can be loaded and used only on OpenVMS AXP systems. However, they can safely reside in a license database (LDB) shared by both OpenVMS VAX and OpenVMS AXP systems.

Because the License Management Facility (LMF) for OpenVMS AXP is capable of handling all types of PAKs, including those for OpenVMS VAX, Digital recommends that you perform your LDB tasks using the OpenVMS AXP LMF.

Availability PAKs for OpenVMS VAX (availability PAKs without the ALPHA option) will not load on OpenVMS AXP systems. Only those availability PAKs containing the ALPHA option will load on OpenVMS AXP systems.

Other PAK types such as activity (also known as concurrent or *n*-user) and personal use (identified by the RESERVE_UNITS option) work on both OpenVMS VAX and OpenVMS AXP systems.

Avoid using the following LICENSE commands from an OpenVMS VAX system on a PAK containing the ALPHA option:

- REGISTER
- DELETE/STATUS
- DISABLE
- ENABLE
- ISSUE
- MOVE
- COPY
- LIST

Caution

By default, all OpenVMS AXP PAKs look disabled to an OpenVMS VAX system. Never use the DELETE/STATUS=DISABLED command from an OpenVMS VAX system on an LDB that contains OpenVMS AXP PAKs. If you do, all OpenVMS AXP PAKs will be deleted.

With the exception of the DELETE/STATUS=DISABLED command, if you inadvertently use one of the LICENSE commands listed previously on an OpenVMS AXP PAK while using an OpenVMS VAX system, the PAK and the database probably will not be affected adversely. Repeat the command using LMF running on an OpenVMS AXP system; the PAK should return to a valid state.

If you fail to repeat the command using LMF on an OpenVMS AXP system, the OpenVMS AXP system will be mostly unaffected. At worst, an OpenVMS AXP PAK that you intended to disable will remain enabled. Only OpenVMS AXP LMF can disable an OpenVMS AXP PAK.

However, if you attempt to use any of the commands listed above on a PAK located in an LDB that is shared with an OpenVMS VAX system, the following serious problems may result:

- Because OpenVMS AXP PAKs look disabled to an OpenVMS VAX system, they are normally ignored at load time by OpenVMS VAX systems. However, if one of the commands listed previously is entered from an OpenVMS VAX system and the PAK information is not set to a valid state by an OpenVMS AXP system, the OpenVMS VAX system may attempt to load the OpenVMS AXP PAK. Because the OpenVMS VAX system will be unable to load the PAK, the OpenVMS VAX LMF will report an error.
- Even if a valid OpenVMS VAX PAK for the affected product is in the LDB, it, too, might not load. In this case, system users might be denied access to the product.

If the PAK cannot be restored to a valid state because all OpenVMS AXP systems are inaccessible for any reason, use your OpenVMS VAX system to disable the OpenVMS AXP. This prevents your VAX system from attempting to load the OpenVMS AXP.

A future release of OpenVMS VAX LMF might remove these command restrictions.

See the *OpenVMS License Management Utility Manual* for more information about using LMF.

2.2.8 PAK Name Difference Using DECnet for OpenVMS AXP

DECnet cluster alias is available on OpenVMS AXP. Note, however, that the PAK name enabling cluster alias routing support on OpenVMS AXP (DVNETEXT) is different from the PAK name enabling cluster alias routing support on OpenVMS VAX (DVNETRTG). The functions supported with the DVNETEXT license differ from the VAX DVNETRTG license. DVNETEXT is supported only to enable level 1 routing on AXP nodes acting as routers for a cluster alias.

Routing between multiple circuits is not supported. Level 2 routing is not supported on DECnet for OpenVMS AXP nodes.

The PAK name for the end node license (DVNETEND) is the same on AXP and VAX systems.

See Chapter 6 for more information about DECnet for OpenVMS AXP.

2.2.9 DECwindows Motif for OpenVMS AXP

The DEC 3000 series workstations display graphics. Part of DECwindows Motif for OpenVMS AXP (DECwindows) software is bundled with the OpenVMS AXP operating system and part is packaged as a layered product. The DECwindows display server, drivers, and fonts are automatically installed when you install OpenVMS AXP. The DECwindows layered product, containing programming and application support, must be installed separately. To run DECwindows applications locally on DEC 3000 series workstations, you must install the DECwindows layered product.

Refer to the *DECwindows Motif for OpenVMS AXP Version 1.1 Installation Guide* for complete instructions.

2.2.10 Multihead Configuration

A DEC 3000 series computer will be configured automatically for multihead use if you rename the private server setup file from a template file type to a command procedure file type. The DECwindows Motif for OpenVMS AXP (DECwindows) server loads this command procedure on startup or restart. Rename this file after installing DECwindows and after logging in to your system.

Note

A multihead configuration consists of a single DEC 3000 series workstation that supports multiple graphics options. A graphics option consists of a graphics controller and a graphics display interface (monitor).

The command procedure always configures the console as the primary head, or screen 0. Note that the firmware always selects the lowest device found in the system (that is, the device with the lowest TURBOchannel slot address) as the console device.

To rename the private server setup file, enter the following command:

```
$ RENAME SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.TEMPLATE
_To: SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM
```

After you have renamed the private server setup file, use the following command to restart the DECwindows server:

```
$ @SYS$STARTUP:DECW$STARTUP RESTART
```

2.2.11 SYSGEN Utility and System Parameters

The OpenVMS AXP System Generation utility (SYSGEN) is available for examining and modifying system parameters on the active system and for examining and modifying the system parameter file ALPHAVMSSYS.PAR.[1] Those functions are similar to the OpenVMS VAX SYSGEN. However, OpenVMS AXP SYSGEN and OpenVMS VAX SYSGEN differ in the following ways:

- OpenVMS AXP includes several new and modified system parameters. Some of the system parameter changes are due to new features. Other changes are due to the larger page sizes of AXP computers. See Section 2.2.11.1 through Section 2.2.11.4 for information about the new system parameters; also see Chapter 5 for information about changes to system parameters.
- On OpenVMS AXP, I/O subsystem configuration capabilities have been removed from SYSGEN. The System Management utility (SYSMAN) provides this functionality on OpenVMS AXP.

  Refer to Section 2.2.12 and to Appendix B for more information about the SYSMAN I/O subsystem configuration commands.

[1] The file name VAXVMSSYS.PAR is used on OpenVMS VAX systems.

2.2.11.1 MULTIPROCESSING System Parameter

The MULTIPROCESSING system parameter controls loading of the OpenVMS system synchronization image, which is used to support symmetric multiprocessing (SMP) options on supported AXP and VAX computers. For OpenVMS AXP Version 1.5, the MULTIPROCESSING system parameter has a new value (4) that is not available on VAX computers with SMP. When MULTIPROCESSING is set to 4, OpenVMS always loads the streamlined multiprocessing synchronization image, regardless of system configuration or CPU availability.

See Section 2.2.13 for more information about SMP.

2.2.11.2 PHYSICAL_MEMORY System Parameter

OpenVMS AXP does not have the PHYSICALPAGES system parameter. Use the system parameter PHYSICAL_MEMORY instead of PHYSICALPAGES. If you want to reduce the amount of physical memory available for use, change the PHYSICAL_MEMORY parameter. The default setting for the PHYSICAL_MEMORY parameter is −1 (unlimited).

2.2.11.3 POOLCHECK System Parameter

The adaptive pool management feature described in Section 5.4 makes use of the POOLCHECK system parameter. The feature maintains usage statistics and extends detection of pool corruption.

Two versions of the SYSTEM_PRIMITIVES executive image are provided that give you a boot-time choice of either a minimal pool-code version or a pool-code version that features statistics and corruption detection:

- POOLCHECK zero (default value)

  SYSTEM_PRIMITIVES_MIN.EXE is loaded.

- POOLCHECK nonzero

  SYSTEM_PRIMITIVES.EXE, pool checking, and monitoring version are loaded.

These features are available on systems running OpenVMS AXP or OpenVMS VAX Version 6.0 and later releases. The features are not available on systems running VAX VMS Version 5.5 and earlier releases.

See Section 5.4 for more information.

2.2.11.4 ITB_ENTRIES and GH_RSRVPGCNT System Parameters

Two new system parameters are associated with the granularity hint regions (GHR) feature described in Section 5.5. They are ITB_ENTRIES and GH_RSRVPGCNT. The ITB_ENTRIES parameter specifies the number of GHRs usable by OpenVMS AXP (default=1). The GH_RSRVPGCNT parameter specifies the number of unused pages within a GHR to be retained after startup (default=0).

Refer to Section 2.2.22 and Section 5.5 for more information.

2.2.12 Using the SYSMAN Utility to Configure the I/O Subsystem

Use the System Management utility (SYSMAN) on OpenVMS AXP computers to connect devices, load I/O device drivers, and debug device drivers. These functions are provided by SYSGEN on OpenVMS VAX computers.

Enter the following command to invoke SYSMAN:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN>
```

Appendix B contains complete format descriptions for the IO AUTOCONFIGURE, IO CONNECT, IO LOAD, IO SET PREFIX, IO SHOW BUS, IO SHOW DEVICE, and IO SHOW PREFIX commands. Table 2–3 compares the I/O subsystem configuration commands on OpenVMS AXP and OpenVMS VAX.

Table 2–3 Comparison of I/O Subsystem Configuration Commands

OpenVMS VAX SYSGEN Command	OpenVMS AXP SYSMAN Command[1]
AUTOCONFIGURE *adapter-spec* or AUTOCONFIGURE ALL.	The default for IO AUTOCONFIGURE is all devices. There is no parameter to the IO AUTOCONFIGURE command. The /SELECT and /EXCLUDE qualifiers are not mutually exclusive, as they are on OpenVMS VAX. Both qualifiers can be specified on the command line.
CONFIGURE.	Used on VAX for Q–bus and UNIBUS, which are not supported on OpenVMS AXP.
CONNECT/ADAPTER requires CMKRNL privilege only.	IO CONNECT requires CMKRNL and SYSLCK privileges.
CONNECT/ADAPTER offers the /ADPUNIT qualifier.	No equivalent.
CONNECT/ADAPTER offers the /CSR_OFFSET qualifier.	Use IO CONNECT/ADAPTER/CSR. Note: CSR is the control and status register.
CONNECT/ADAPTER offers the /DRIVERNAME (no underscore) qualifier.	IO CONNECT offers the /DRIVER_NAME qualifier.
No equivalent.	IO CONNECT offers the /LOG=(ALL,CRB,DDB,DPT,IDB,SC,UCB) qualifier and options.
CONNECT/ADAPTER offers the /MAXUNITS (no underscore) qualifier.	IO CONNECT offers the /MAX_UNITS qualifier.
No equivalent.	IO CONNECT offers the /NUM_UNITS qualifier.
CONNECT/ADAPTER offers the /NUMVEC (no underscore) qualifier.	IO CONNECT offers the /NUM_VEC qualifier.
CONNECT/ADAPTER uses the /SYSIDHIGH and /SYSIDLOW qualifiers.	IO CONNECT provides the /SYS_ID qualifier to indicate the SCS system ID of the remote system to which the device is to be connected.

[1] All I/O subsystem configuration commands on OpenVMS AXP are preceded by "IO".

(continued on next page)

Table 2–3 (Cont.) Comparison of I/O Subsystem Configuration Commands

OpenVMS VAX SYSGEN Command	OpenVMS AXP SYSMAN Command[1]
CONNECT/ADAPTER provides the /VECTOR_OFFSET qualifier to specify the offset *from* the interrupt vector address of the multiple device board *to* the interrupt vector address for the specific device being connected.	No equivalent.
No equivalent.	IO CONNECT provides the /VECTOR_SPACING qualifier.
CONNECT CONSOLE.	OpenVMS AXP does not require this command.
LOAD requires CMKRNL privilege.	IO LOAD requires CMKRNL and SYSLCK privileges. Also, IO LOAD provides the /LOG=(ALL,DPT) qualifier to display information about drivers that have been loaded.
RELOAD.	Not supported.
No equivalent.	IO SET PREFIX sets the prefix list used to manufacture the IOGEN Configuration Building Module (ICBM) names.
SHOW/ADAPTER.	Use IO SHOW BUS, which lists all the buses, node numbers, bus names, TR numbers, and base CSR addresses.
SHOW/CONFIGURATION.	Used on VAX for Q–bus and UNIBUS, which are not supported. Use IO SHOW BUS.
SHOW/DEVICE displays full information about the device drivers loaded into the system, including the start and end address of each device driver.	The command is IO SHOW DEVICE. Start and end address information is not shown.
SHOW/DRIVER displays the start and end addresses of device drivers loaded into the system.	The command is IO SHOW DRIVER. It displays the loaded drivers but does not display the start and end addresses because drivers may be loaded into granularity hint regions.
No equivalent.	IO SHOW PREFIX displays the current prefix list used in the manufacture of ICBM names.
SHOW/UNIBUS.	No equivalent; UNIBUS devices are not supported on AXP processors.

[1] All I/O subsystem configuration commands on OpenVMS AXP are preceded by "IO".

First, you should familiarize yourself with the differences between the I/O subsystem configuration commands in OpenVMS VAX SYSGEN and OpenVMS AXP SYSMAN. Next, change the DCL procedures (if you copied any over from the VAX to the AXP system) that include commands such as:

```
$ SYSGEN :== $SYS$SYSTEM:SYSGEN
$ SYSGEN io-subsystem-configuration-command
```

to:

```
$ SYSMAN :== $SYS$SYSTEM:SYSMAN
$ SYSMAN IO io-subsystem-configuration-command
```

Look for differences in the command parameters and qualifiers, as noted in Table 2–3.

Note

For OpenVMS AXP, SYSMAN IO AUTOCONFIGURE occurs automatically at startup.

2.2.13 Symmetric Multiprocessing on AXP Systems

Symmetric multiprocessing (SMP) is supported on OpenVMS AXP Version 1.5. Refer to the OpenVMS AXP *Software Product Description* (SPD 41.87.*xx*) for the most up-to-date information about supported SMP configurations.

On the supported AXP systems, SMP is enabled automatically by the console firmware as long as there are multiple CPUs and the environment variable *cpu_enabled* is set either to *ff hex* or to the mask of available CPUs. (Each bit corresponds to a CPU. For example, bit 0 corresponds to CPU 0, and so forth.)

SMP also is managed on AXP and VAX systems by using the MULTIPROCESSING system parameter. MULTIPROCESSING controls the loading of the system synchronization image. The system parameter's values of 0, 1, 2, and 3 have equivalent functions on AXP and VAX systems; however, the value 4 is an option specific on AXP systems. Table 2–4 summarizes the functions of the five MULTIPROCESSING values.

Table 2–4 MULTIPROCESSING Values on AXP and VAX Systems

Value	Function
0	Load uniprocessing synchronization image.
1	Load full-checking multiprocessing synchronization image if CPU type is capable of SMP and two or more CPUs are present on the system.
2	Always load full-checking version, regardless of system configuration or CPU availability.
3	Load streamlined multiprocessing synchronization image if CPU type is capable of SMP and two or more CPUs are present on the system.
4	On OpenVMS AXP systems, always load streamlined multiprocessing synchronization image, regardless of system configuration or CPU availability.

When the full-checking multiprocessing synchronization image is loaded, OpenVMS performs software sanity checks on the node's CPUs; also, OpenVMS provides a full history of CPU information in the event of a system failure. OpenVMS stores a program counter (PC) history in the spinlock (SPL) structures used to synchronize system activity. When the system fails, that information is accessible by using the SDA command SHOW SPINLOCK. The information displayed includes the PCs of the last 16 acquisitions and releases of the spin locks.

The performance of an SMP node running the full-checking image is slower compared with a node running the streamlined image. However, it is easier to debug failures on SMP nodes (if you are writing privileged code) when the full-checking image is enabled. The streamlined image is designed for faster performance, with a trade-off of less extensive debug support following a system failure.

In addition to MULTIPROCESSING, the following system parameters control the behavior of an SMP system. These parameters have equivalent functions on AXP and VAX multiprocessing systems.

- SMP_CPUS system parameter

  SMP_CPUS identifies which secondary processors, if available, are to be booted into the multiprocessing system at boot time. SMP_CPUS is a 32-bit mask; if a bit in the mask is set, the processor with the corresponding CPU ID is booted into the multiprocessing system (if it is available). For example, if you want to boot only the CPUs with CPU IDs 0 and 1, specify the value 3 (both bits are on). The default value of SMP_CPUS, −1, boots all available CPUs into the multiprocessing system.

  Although a bit in the mask corresponds to the primary processor's CPU ID, the primary processor is always booted. That is, if the mask is set to 0, the primary CPU will still boot. Any available secondary processors will not be booted into the multiprocessing system.

  The SMP_CPUS system parameter is ignored if the MULTIPROCESSING parameter is set to 0.

- SMP_LNGSPINWAIT system parameter

  Certain shared resources in a multiprocessing system take longer to become available than allowed for by the SMP_SPINWAIT parameter. The SMP_LNGSPINWAIT parameter establishes, in 10-microsecond intervals, the amount of time a CPU in an SMP system waits for these resources. A timeout causes a CPUSPINWAIT bugcheck. For SMP_LNGSPINWAIT, the default value of 300,000 10-microsecond intervals (3 seconds) is usually adequate.

- SMP_SANITY_CNT system parameter

  SMP_SANITY_CNT establishes, in 10-millisecond clock ticks, the timeout interval for each CPU in a multiprocessing system. Each CPU in an SMP system monitors the sanity timer of one other CPU in the configuration to detect hardware or software failures. If allowed to go undetected, these failures could cause the cluster to hang. A timeout causes a CPUSANITY bugcheck. For SMP_SANITY_CNT, the default value of 30 10-millisecond intervals (300 milliseconds) is usually adequate.

- SMP_SPINWAIT system parameter

  SMP_SPINWAIT establishes, in 10-microsecond intervals, the amount of time a CPU normally waits for access to a shared resource. This process is called **spinwaiting**. A timeout causes a CPUSPINWAIT bugcheck. For SMP_SPINWAIT, the default value of 10,000 10-microsecond intervals (100 milliseconds) is usually adequate.

The output of MONITOR/MODE=CPU commands on AXP and VAX systems contains the same type of information.

The SHOW CPU command displays information about the status, characteristics, and capabilities of the processors active in and available to an OpenVMS multiprocessing system. The display is the same for SHOW CPU/BRIEF commands on AXP and VAX systems running SMP.

However, when executed on an AXP system, the SHOW CPU/FULL command output contains information not found in the display from a VAX SMP node. In the following VAX example, the SHOW CPU/FULL command produces a configuration summary of the VAX 6000-420 system OLEO, indicating that only CPU 02, the primary CPU, is active and in the RUN state. It also shows that there is a uniprocessing driver loaded in the system, thus preventing the system from being enabled as a multiprocessor.

```
$  ! On a VAX system
$  SHOW CPU/FULL

OLEO, A VAX 6000-420
Multiprocessing is DISABLED. MULTIPROCESSING Sysgen parameter = 02
Minimum multiprocessing revision levels -- CPU: 0 uCODE: 0 UWCS: 21.
PRIMARY CPU = 02
*** Loaded unmodified device drivers prevent multiprocessor operation.***
    RBDRIVER

CPU 02 is in RUN state
Current Process: Koko               PID = 2A6001E3
Revision levels: CPU: 0 uCODE: 0 UWCS: 0.
Capabilities of this CPU:
        PRIMARY   VECTOR RUN
Processes which can only execute on this CPU:
        CONFIGURE         PID = 2A40010B  Reason = PRIMARY Capability
                                          Reason = RUN Capability

CPU 07 is in INIT state
Current Process: *** None ***
Revision levels: CPU: 0 uCODE: 0 UWCS: 0.
Capabilities of this CPU:
    *** None ***
Processes which can only execute on this CPU:
    *** None ***
```

In comparison, the following SHOW CPU/FULL display is from a four-CPU AXP system:

```
$  ! On an AXP system
$ SHOW CPU/FULL

CPU type: DEC 7000 Model 640
Multiprocessing is ENABLED. Full checking synchronization image loaded.
Minimum multiprocessing revision levels: CPU = 1
System Page Size = 8192

System Revision Code = System Serial Number = PROTO115
Default CPU Capabilities:
 QUORUM RUN
Default Process Capabilities:
 QUORUM RUN
PRIMARY CPU = 00
CPU 00 is in RUN state
Current Process: *** None ***
```

```
Serial Number: GROUCHO
Revision:
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.37
PALcode Compatibility = 3
Maximum Shared Processors = 8
Memory Space: Physical address = 00000000 00000000
Length = 16
Scratch Space: Physical address = 00000000 00020000
Length = 16
Capabilities of this CPU:
 PRIMARY QUORUM RUN
Processes which can only execute on this CPU:
 CONFIGURE        PID = 00000024  Reason: PRIMARY Capability
CPU 01 is in RUN state
Current Process: _RTA1:          PID = 0000002E

Serial Number: HARPO
Revision:
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.37
PALcode Compatibility = 3
Maximum Shared Processors = 8
Memory Space: Physical address = 00000000 00000000
Length = 16
Scratch Space: Physical address = 00000000 00020000
Length = 16
Capabilities of this CPU:
 QUORUM RUN
Processes which can only execute on this CPU:
        *** None ***
CPU 02 is in RUN state
Current Process: *** None ***

Serial Number: CHICO
Revision:
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.37
PALcode Compatibility = 3
Maximum Shared Processors = 8
Memory Space: Physical address = 00000000 00000000
Length = 16
Scratch Space: Physical address = 00000000 00020000
Length = 16
Capabilities of this CPU:
 QUORUM RUN
Processes which can only execute on this CPU:
        *** None ***
CPU 03 is in RUN state
Current Process: *** None ***
```

```
Serial Number: ZEPPO
Revision:
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.37
PALcode Compatibility = 3
Maximum Shared Processors = 8
Memory Space: Physical address = 00000000 00000000
Length = 16
Scratch Space: Physical address = 00000000 00020000
Length = 16
Capabilities of this CPU:
 QUORUM RUN
Processes which can only execute on this CPU:
        *** None ***
```

The console PALcode revision level numbers on AXP systems might be different from the numbers shown in the previous example.

2.2.14 Startup Command Procedures

As a result of the SYSMAN IO commands described in Section 2.2.12 and Appendix B, you might need to modify some of your existing SYS$STARTUP:*.COM procedures *if you copy them to an OpenVMS AXP system disk.* Note, however, that the command procedures provided by OpenVMS AXP have been modified to invoke SYSMAN, instead of SYSGEN, for I/O subsystem configuration commands.

Search for AUTOCONFIGURE and update the associated command interface. For example:

```
$ SEARCH SYS$STARTUP:*.COM AUTOCONFIGURE
```

Change SYSGEN AUTOCONFIGURE [ALL] to SYSMAN IO AUTOCONFIGURE.

2.2.15 Devices on OpenVMS AXP

Refer to the OpenVMS AXP *Software Product Description* (SPD 41.87.*xx*) for the most up-to-date information about the hardware devices supported with the available AXP computers.

2.2.16 Local DSA Device Naming

On OpenVMS AXP, all local Digital Storage Architecture (DSA) devices use a controller letter of A, regardless of the physical controller on which the device resides. All local DSA disk devices are named DUA*n* or DJA*n*, where *n* is the unique disk unit number. All local DSA tape devices are named MUA*n*, where *n* is the unique tape unit number.

The OpenVMS AXP local device-naming scheme represents a change from OpenVMS VAX, where local DSA devices inherit the controller letter from the physical controller on which the device resides.

Table 2–5 compares the new OpenVMS AXP local DSA device-naming scheme with the local naming schemes on OpenVMS VAX and the DEC 7000 Model 600 AXP console. Note that the DEC 7000 Model 600 AXP console uses the OpenVMS VAX local DSA device-naming scheme when referring to local DSA devices. As a result, you must specify the OpenVMS VAX local DSA device names when you use the DEC 7000 Model 600 AXP console commands BOOT and SHOW DEVICE.

Table 2–5 Comparison of Device Naming on OpenVMS

Controller Where Disk Resides	OpenVMS VAX and DEC 7000 Model 600 AXP Console Local Device Naming	OpenVMS AXP Local Device Naming
PUA0	DUA0	DUA0
PUB0	DUB14	DUA14
PUC0	DUC115	DUA115

As shown in Table 2–5, OpenVMS VAX names disk unit 14 on controller PUB0 as DUB14, while OpenVMS AXP names this unit DUA14. On OpenVMS AXP, use of a single controller letter requires that the unit number for each local DSA device be unique.

Controller letters are used in device naming for hardware that artificially restricts unit number ranges. For example, Small Computer Systems Interface (SCSI) controllers currently can have disk unit numbers only from 0 through 7, which almost precludes sufficient uniqueness for any large system requiring many disks. By contrast, current DSA disks have a unit number range of 0 through 4000. In addition, the allocation class can be used to differentiate device names further. As a result, the OpenVMS AXP operating system does not add uniqueness to the device name via the controller letter.

The following benefits result from the change in local DSA device naming:

- Device naming is more uniform. Local DSA device naming is now identical to the scheme used for local DSSI devices and remote DSA devices.
- System management is simplified. Because all DSA devices now have unique unit numbers, an operator can unambiguously locate a device from among a system's disks using only the device's unit number. The operator need not be concerned whether a device with unit number 0 is DUA0 or DUB0.
- Dual pathing of a device between two OpenVMS AXP systems with local controllers (unsupported in OpenVMS AXP) is easier. Dual pathing is possible only if the device is named identically throughout the VMScluster.

  On OpenVMS VAX, the device name inherits the controller letter from the controller on which the device resides. You must take great care to place the device on identically named controllers in each OpenVMS VAX system so that the resulting device names are identical.

  With the OpenVMS AXP local DSA-naming scheme, device names are not sensitive to the controller on which the device resides, and the names always use a controller letter of A. Dual pathing can be configured without regard to the local controller on which the dual-pathed device resides.

The change in local DSA device naming in OpenVMS AXP may require that you make some changes. If local DSA devices are not already unique by unit number, you might need to reconfigure DSA devices when moving from OpenVMS VAX to OpenVMS AXP. Local DSA physical device names that are hardcoded in command files or applications may also be affected by this change.

2.2.17 ISO 9660 Standard and Restrictions on OpenVMS AXP

This section describes problems and restrictions that apply to OpenVMS support of the ISO 9660 standard. This information, presented in the *OpenVMS AXP Version 1.5 Release Notes*, is repeated here as a convenience to system managers.

2.2.17.1 Volume Labels

For ISO 9660 media, volume labels can contain from 1 to 32 characters. The first 12 characters are used to produce a unique volume identity. If the label is not unique within the first 12 characters, the volume will not mount and the following error message is displayed:

```
%SYSTEM-F-VOLALRMNT, another volume of the same label already mounted
```

To resolve this problem, mount the volume specifying a different volume label, and use the /OVERRIDE=IDENTIFIER command qualifier.

2.2.17.2 Volume Set Names

An ISO 9660 volume set name can be from 1 to 128 characters in length. The first 12 characters are used to produce a unique volume set identity. If the volume set name is not unique within the first 12 characters, the volume will not mount and the following error message is displayed:

```
%SYSTEM-F-VOLINSET, volume is already part of another volume set
```

To resolve this problem, mount the volume specifying a new volume set name with the /BIND=*volume-set-name* command qualifier.

2.2.17.3 Volume Set and Volume Set Name Duplication

The first 12 characters of both the volume set and the volume set name are used to produce different lock manager resource names, which are then used to coordinate volume and volume set associations. If both the volume name and the volume set name are the same (within the first 12 characters), a lock manager deadlock error occurs and the following error message is displayed:

```
%SYSTEM-F-DEADLOCK, deadlock detected
```

To resolve this problem, mount the volume specifying a different volume label, and use the /OVERRIDE=IDENTIFIER command qualifier.

2.2.17.4 InfoServer Served Volumes

The client software that makes InfoServer served volumes available only recognizes volumes whose media format is Files–11 On-Disk Structure Level 2 (ODS-2). To make ISO 9660 or High Sierra volumes visible to client software and accessible to client nodes, use the following command on the InfoServer for each ISO 9660 or High Sierra volume:

```
INFOSERVER> create service BOOKREADER for DUA0: class ODS-2
```

2.2.17.5 Undefined Record Format Errors

Many ISO 9660 CD–ROMs are mastered without a specified record format because the ISO 9660 media can be mastered from platforms that do not support the semantics of files containing predefined record formats.

OpenVMS file system utilities (such as TYPE and COPY), language RTLs, and applications that use RMS for record access may report RMS errors, utility errors, and language errors when accessing files whose record format is undefined or appears illegally specified.

To avoid this problem, use the following command syntax at mount time to force all files of type UNDEFINED to the STREAM record format having a maximum record length of 512 bytes:

```
MOUNT/MEDIA=CD-ROM/UNDEFINED=(STREAM:512) device label
```

For more information about RMS record formatting, see the *OpenVMS Record Management Utilities Reference Manual* and the *OpenVMS Record Management Services Reference Manual.*

2.2.17.6 PATHWORKS Access to ISO 9660 in a WAN or LAN Environment

To access ISO 9660 volumes in a wide area network (WAN) or local area network (LAN) environment, issue a MOUNT/SYSTEM command for the ISO 9660 volume.

On the personal computer (PC) client node, assign the volume to a PC device using the appropriate command. For example, in an MS–DOS environment, the assignment command might look like the following:

```
B:> USE ?: \\MYNODE\DISK$VOLUME:[000000]%VMSUSER *
```

where:

- USE ?: commands the USE utility to assign the next available PC device.
- \ \MYNODE indicates an OpenVMS node.
- \DISK$VOLUME:[000000] indicates the volume and directory.
- %VMSUSER * indicates the access control string. The asterisk (*) causes MS–DOS to prompt you for the password.

2.2.18 File Name Format of Drivers Supplied by Digital on AXP

All drivers supplied by Digital on OpenVMS AXP use the following format:

```
facility-name$xxDRIVER.EXE
```

The drivers included on the OpenVMS AXP kit use SYS for *facility-name.*

On OpenVMS VAX, no facility prefix is present or permitted for drivers. They are simply named *xx*DRIVER.EXE.

Note

User-written device drivers are not supported on OpenVMS AXP systems.

2.2.19 OpenVMS Installation Media

OpenVMS AXP operating system binaries and documentation are distributed on a compact disc. Other media for installations are not available. See Section C.3 for information about providing users access to the online documentation.

2.2.20 VMSINSTAL Utility

OpenVMS AXP provides a new version of the VMSINSTAL utility. This version contains new callbacks, in addition to changes and new features for existing callbacks. Software developers at your site who are creating OpenVMS based software kits should read the *OpenVMS Developer's Guide to VMSINSTAL* for details.

The following new VMSINSTAL utility features are of interest to system managers:

- History file of VMSINSTAL executions
- Product installation log file
- Procedure for listing installed products

2.2.20.1 History File of VMSINSTAL Executions

When VMSINSTAL terminates, a history file records the name of the product being installed and the status of the attempted installation. The history file is named SYS$UPDATE:VMSINSTAL.HISTORY.

2.2.20.2 Product Installation Log File

If a product installation is successful using VMSINSTAL, a log file is created. This file contains information indicating:

- The product that was installed
- Who installed the product
- What files were added, deleted, modified, and so on

This file is created as SYS$UPDATE:*facvvu*.VMI_DATA.

2.2.20.3 Procedure for Listing Installed Products

A new procedure, SYS$UPDATE:INSTALLED_PRDS.COM, lets the user check what products have been installed. The procedure has an optional parameter for indicating a restricted search of installed products. When executed, this procedure lists the product's name and version, when it was installed, and who installed it.

The command format is as follows:

@SYS$UPDATE:INSTALLED_PRDS   [product-mnemonic]

The *product-mnemonic* value is optional. To use it, specify the save-set name of the product. If you specify *product-mnemonic*, only log files belonging to the specified product will have installation data displayed. The product mnemonic can be passed to the procedure by using any of the following search criteria:

- Product name and version (save-set name)
- Product name only
- Wildcards

The following command examples illustrate the installed products procedure using the search criteria:

```
$ @SYS$UPDATE:INSTALLED_PRDS

$ @SYS$UPDATE:INSTALLED_PRDS DTR010

$ @SYS$UPDATE:INSTALLED_PRDS DTR

$ @SYS$UPDATE:INSTALLED_PRDS DTR*
```

2.2.21 Running the AUTOGEN Procedure

AUTOGEN is included with OpenVMS AXP. Use it to adjust the values of system parameters after installing OpenVMS AXP and after installing layered products.

Note

The VAXVMSSYS.PAR system parameter file on OpenVMS VAX systems is called ALPHAVMSSYS.PAR on OpenVMS AXP. Like VAXVMSSYS.PAR, the ALPHAVMSSYS.PAR file resides in the SYS$SYSTEM directory.

The following notes apply to AUTOGEN:

- Feedback mode is supported. Follow the recommendations and procedures for using feedback mode (described in the *OpenVMS System Manager's Manual*) to adjust your system parameters according to the system's work load. After at least 24 hours of system operation, the system manager can execute the following command to save current feedback data:

  ```
  $ RUN SYS$SYSTEM:AGEN$FEEDBACK
  ```

  You can use this command any time, and you can repeat it as system operation time increases. The saved feedback data is used subsequently by AUTOGEN if the starting phase of GETDATA is specified.

- AUTOGEN increases allocations where indicated unless an exact or maximum value is specified in the MODPARAMS.DAT file.

- The AGEN$PARAMS.REPORT file contains additional information. A copy of the parameters found during the GETDATA phase and the final setting of parameters determined during GENPARAMS are included along with informational, advisory, and warning messages.

- Some system parameters are in units of pagelets, whereas others are in units of pages. AUTOGEN determines the hardware page size and records it in the PARAMS.DAT file. When reviewing AUTOGEN recommended values or when setting system parameters in MODPARAMS.DAT, note carefully which units are required for each parameter.

  See Section 5.1 for information about system parameters and their units and about the tuning considerations.

2.2.22 Improving the Performance of Main Images and Shareable Images

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

Also, the OpenVMS operating system executive images are, by default, loaded into GHRs. The result is an improvement in overall OpenVMS system performance.

These options are not available on OpenVMS VAX systems.

The GHR feature lets OpenVMS split the contents of images and sort the pieces so that they can be placed with other pieces that have the same page protection in the same area of memory. Consequently, TBs on AXP systems are used more efficiently than if the loadable executive images or a user's main image or shareable images were loaded in the traditional manner.

See Section 5.5 and Section A.2.5.4 for details.

2.2.23 SYS.EXE Renamed to SYS$BASE_IMAGE.EXE

The loadable executive image SYS.EXE has been renamed SYS$BASE_IMAGE.EXE.

By renaming the loadable executive image SYS.EXE to SYS$BASE_IMAGE.EXE, the name conforms to OpenVMS naming standards, which state that SYS$ must be prefixed to the image name.

2.2.24 Security Audit Log File Name

The security audit log file, which resides in SYS$COMMON:[SYSMGR], is called SECURITY.AUDIT$JOURNAL on OpenVMS VAX Version 6.0. The file is called SECURITY_AUDIT.AUDIT$JOURNAL on OpenVMS AXP Version 1.5 and on VMS Version 5.5 and earlier releases.

See Chapter 4 for information about security differences between OpenVMS VAX Version 6.0, which supports C2 level security, and OpenVMS AXP Version 1.5.

2.2.25 SYSUAF.DAT File and Process Limits and Quotas

The Authorize utility commands and parameters are identical. However, the default values for a number of OpenVMS AXP process limits and quotas are higher.

As you know, you can use the DEFAULT account values as a template for new accounts. The values used for the DEFAULT account also are the default values for unspecified process limits and quotas when you use the ADD command to create a new account.

Table 2–6 lists the changes to the DEFAULT account process limit and quota values.

Note

For the quota values that are in pages (on VAX) or pagelets (on AXP), remember that *each page or pagelet* represents the same 512-byte quantity.

Also, see Section 2.2.26 for special considerations when using a common SYSUAF.DAT file in a dual-architecture VMScluster.

Table 2–6 Comparison of Default Values for Process Limits and Quotas

Limit or Quota	VMS Version 5.5 Value	OpenVMS VAX Version 6.0 Value	OpenVMS AXP Version 1.5 Value	Description
ASTLM	24	40	250	Maximum number of asynchronous system trap (AST) operations and scheduled wake-up requests that the process can have queued at one time.
BIOLM	18	40	150	Maximum number of buffered I/O operations (such as terminal I/O) that the process can have outstanding at one time.
BYTLM	8192 bytes	32768 bytes	64000 bytes	Maximum number of bytes of nonpaged system dynamic memory that the process's job can consume at one time.
CPUTIME	0	0	0	CPU time limit (0 = no limit).
DIOLM	18	40	150	Maximum number of direct I/O operations (usually disk) that the process can have outstanding at one time.
ENQLM	100	200	2000	Maximum number of locks that the process can have queued at one time.
FILLM	20	300	100	Open file limit.
JTQUOTA	1024 bytes	4096 bytes	4096 bytes	Initial byte quota with which the jobwide logical name table (for this process) is to be created.
PGFLQUOTA	10240 VAX pages	32768 VAX pages	50000 AXP pagelets	Maximum number of pages or pagelets that the process can use in the system paging file. Each VAX page equals 512 bytes. Each AXP pagelet also equals 512 bytes, or one-sixteenth of a full page on an AXP computer with 8KB pages; thus, 50000 pagelets equal 3125 AXP pages.
PRCLM	2	2	8	Maximum number of subprocesses that can exist at one time for the process.
WSDEFAULT	150 VAX pages	256 VAX pages	2000 AXP pagelets	Initial limit to the number of physical pages or pagelets that the process can use. Each VAX page equals 512 bytes. Each AXP pagelet also equals 512 bytes, or one-sixteenth of a full page on an AXP computer with 8KB pages; thus, 2000 pagelets equal 125 AXP pages.

(continued on next page)

Table 2–6 (Cont.) Comparison of Default Values for Process Limits and Quotas

Limit or Quota	VMS Version 5.5 Value	OpenVMS VAX Version 6.0 Value	OpenVMS AXP Version 1.5 Value	Description
WSEXTENT	512 VAX pages	1024 VAX pages	16384 AXP pagelets	Maximum amount of physical memory allowed to the process. Each VAX page equals 512 bytes. Each AXP pagelet also equals 512 bytes, or one-sixteenth of a full page on an AXP computer with 8KB pages; thus, 16384 pagelets equal 1024 AXP pages.
WSQUOTA	256 VAX pages	512 VAX pages	4000 AXP pagelets	Maximum amount of physical memory a user process can lock into its working set. WSQUOTA value also represents the maximum amount of swap space that the system reserves for the process, and the maximum amount of physical memory that the system allows the process to consume if the systemwide memory demand is significant. Each VAX page equals 512 bytes. Each AXP pagelet also equals 512 bytes, or one-sixteenth of a full page on an AXP computer with 8KB pages; thus, 4000 pagelets equal 250 AXP pages.

Because of the broad range of commercial applications that run on VAX and AXP computers, it is difficult to offer meaningful, precise guidelines on process limit and quota values. As an experienced OpenVMS system manager, you already know that the process limit and process quota default values (on both VAX and AXP computers) are only a starting point for your evaluation.

The values that are appropriate for processes on your VAX and AXP computers will be determined by your experimentation and modifications over time. Factors in your decisions about appropriate limit and quota values for each process will include the amount of available memory, CPU processing power, the average work load of the applications, and peak work loads of the applications.

If you copy a SYSUAF.DAT file from a VAX computer to an AXP computer, consider the following:

- The values for process limits and quotas on the OpenVMS AXP probably should not be less than the default values shown in Table 2–6.
- If process limits and process quotas for accounts on the VAX computers were higher than the new AXP default values, do not lower the process limit and quota values to the default values.
- The process limits and process quotas listed in Table 2–6 are higher on AXP computers for several reasons:
  - In the case of memory-related quotas, the increase is to avoid constraining the performance of processes on computers with large amounts of memory.

Translated images and native AXP images, generally speaking, will be larger on AXP systems; processes might need the additional memory that is available on AXP computers to operate efficiently.

The default process values were increased for OpenVMS VAX Version 6.0; for earlier VMS releases, though, keep in mind that the default values were established when VAX computers had smaller amounts of available memory, such as 4MB. (The higher default values for OpenVMS AXP processes might be appropriate defaults for processes on VAX computers with large amounts of available memory.)

- For limits related to queued activities, the default process limit values on OpenVMS VAX computers might be inadequate for many commercial applications on AXP computers. (Again, the same might be true for commercial applications on VAX computers.)

Be careful when you assign and read the OpenVMS AXP SYSUAF process quotas that have values in pagelets (WSDEFAULT, WSQUOTA, WSEXTENT, and PGFLQUOTA). OpenVMS AXP utilities accept and display these quota values in pagelets, and then round up (if warranted). Rounding up occurs on an AXP computer with 8KB pages when the value you specify is not a multiple of 16.

For example, assume that you assign 2100 pagelets to the WSDEFAULT value for a process. On an AXP computer with 8KB pages, 2100 pagelets equal 131.25 AXP pages. The result is that AXP rounds up to 132 AXP pages. Thus, specifying 2100 pagelets is effectively the same as specifying a value in the range of 2096 to 2112 pagelets.

The AXP page-rounding operation can create interesting scenarios for system managers.

- Scenario 1

  You attempt to increase slightly or decrease slightly a process quota in pagelets; in fact, no change in the number of AXP pages allocated for the process occurs internally.

- Scenario 2

  You increase or decrease a process quota in terms of pagelets to a greater extent than you realized.

Scenario 1

Assume that you choose to increase slightly the WSDEFAULT value for a process. The current value is 1985 AXP pagelets, and you increase the value by 10 pagelets to 1995 pagelets. On an AXP computer with 8KB pages, 1985 pagelets equals 124.0625 AXP pages, which is rounded up internally to 125 AXP pages. The new, higher value of 1995 pagelets equals 124.6875 AXP pages, which results in the same 125 AXP pages. The net effect is that an additional working set default size was not allocated to the process, despite the command that increased the value by 10 pagelets.

Scenario 2

Assume that the PGFLQUOTA value for a process is 50000 pagelets. On an AXP computer with 8KB pages, 50000 pagelets equals 3125 AXP pages, or 25,600,000 bytes (3125 pages * 8192 bytes per page). Suppose you enter a modest increase of 10 pagelets, specifying a new PGFLQUOTA value of 50010 pagelets. On an AXP computer with 8KB pages, the 50010 pagelets equals 3125.625 AXP pages, which is rounded up to 3126 AXP pages. The 3126 AXP pages equals 25,608,192 bytes.

While you might have expected the increase of 10 pagelets to result in an additional 5120 bytes for the process PGFLQUOTA, the actual increase was 8192 bytes. The amount of the increase when AXP page boundaries are crossed would be even greater on AXP computers with 16KB, 32KB, or 64KB pages.

2.2.26 How Process Quotas Are Determined in Dual-Architecture VMSclusters

Process quota default values in SYSUAF.DAT on AXP systems are higher than the SYSUAF.DAT defaults on VAX systems. How, then, do you choose values for processes that could run on AXP systems or on VAX systems in a VMScluster? Understanding how a process is assigned quotas when the process is created in a dual-architecture VMScluster environment will help you manage this task.

The quotas to be used by a new process are determined by the OpenVMS LOGINOUT software. LOGINOUT works the same on OpenVMS AXP and VAX VMS systems. When a user logs in and a process is started, LOGINOUT uses the *larger* of:

- The value of the quota defined in the process's SYSUAF.DAT record
- The *current* value of the corresponding PQL_M*quota* system parameter on the host node in the VMScluster

For example, LOGINOUT would compare the value of the account's ASTLM process limit (as defined in the common SYSUAF.DAT) with the value of the PQL_MASTLM system parameter on the host AXP system or on the host VAX system in the VMScluster.

The letter M in PQL_M means minimum. The PQL_M*quota* system parameters set a minumum value for the quotas. In the Current and Default columns of the following edited SYSMAN display, note how the current value of each PQL_M*quota* parameter exceeds its system-defined default value in most cases:

```
SYSMAN> PARAMETER SHOW/PQL

%SYSMAN-I-USEACTNOD, a USE ACTIVE has been defaulted on node FLASHR
Node FLASHR:   Parameters in use: ACTIVE
Parameter Name             Current   Default   Minimum   Maximum Unit  Dynamic
--------------             -------   -------   -------   ------- ----  -------
PQL_MASTLM                     120         4        -1        -1 Ast         D
PQL_MBIOLM                     100         4        -1        -1 I/O         D
PQL_MBYTLM                  100000      1024        -1        -1 Bytes       D
PQL_MCPULM                       0         0        -1        -1 10Ms        D
PQL_MDIOLM                     100         4        -1        -1 I/O         D
PQL_MFILLM                     100         2        -1        -1 Files       D
PQL_MPGFLQUOTA               65536      2048        -1        -1 Pagelets    D
PQL_MPRCLM                      10         0        -1        -1 Processes   D
PQL_MTQELM                       0         0        -1        -1 Timers      D
PQL_MWSDEFAULT                2000      2000        -1        -1 Pagelets
PQL_MWSQUOTA                  4000      4000        -1        -1 Pagelets    D
PQL_MWSEXTENT                 8192      4000        -1        -1 Pagelets    D
PQL_MENQLM                     300         4        -1        -1 Locks       D
PQL_MJTQUOTA                     0         0        -1        -1 Bytes       D
```

In this display, the values for many PQL_M*quota* parameters increased from the defaults to their current values. Typically, this happens over time when AUTOGEN FEEDBACK is run periodically on your system. The PQL_M*quota* values also can change, of course, when you modify the values in MODPARAMS.DAT or in SYSMAN. As you consider the use of a common SYSUAF.DAT in a dual-architecture VMScluster, keep the dynamic nature of the PQL_M*quota* parameters in mind.

The use of a common SYSUAF.DAT in a dual-architecture VMScluster presents interesting scenarios:

- Scenario 1

  The common SYSUAF.DAT file has values that are usually associated with a SYSUAF.DAT from an AXP system, and the process is being created on a VAX node in the VMScluster.

- Scenario 2

  The common SYSUAF.DAT file has values that are usually associated with a SYSUAF.DAT from a VAX system, and the process is being created on an AXP node in the VMScluster.

- Scenario 3

  The common SYSUAF.DAT file has values that are usually associated with a SYSUAF.DAT from an AXP system, and the process is being created on an AXP node in the VMScluster.

- Scenario 4

  The common SYSUAF.DAT file has values that are usually associated with a SYSUAF.DAT from a VAX system, and the process is being created on a VAX node in the VMScluster.

Note

Your task of selecting values for a common SYSUAF.DAT in a dual-architecture VMScluster is similar to the task of selecting common SYSUAF.DAT values for either:

- VAXclusters that include large VAX systems and small VAX systems; for example, VAX 9000 systems and VAXstation 4000 systems.
- Homogeneous AXP VMSclusters that include large AXP systems and small AXP systems; for example, DEC 7000 systems and DEC 3000 systems.

Scenario 1: AXP Level Values in Common SYSUAF, VAX Host Node

Assume that a common SYSUAF.DAT file is used in a dual-architecture VMScluster and that the cluster manager elects to use process quotas equal to the default values in an AXP SYSUAF.DAT file. As shown in Table 2–6, the process default values are higher on AXP systems (as compared with VAX VMS Version 5.5–2 systems). A user logs in to the VMScluster and the J_SMITH process is about to be created *on a VAX system.*

The ASTLM quota sets the maximum number of asynchronous system trap (AST) operations and scheduled wake-up requests that the process can have queued at one time. Assume that the ASTLM value of 250 is defined in the common SYSUAF.DAT record for J_SMITH. This value is the one used by default in AXP SYSUAF.DAT files. When the J_SMITH process is created on the VAX system, LOGINOUT compares the ASTLM process quota value (250) with the active, corresponding PQL_MASTLM system parameter value on the host VAX system. Although the default PQL_MASTLM value on VAX systems is 4, assume that the current PQL_MASTLM value on the host VAX system is 100. Because LOGINOUT uses the larger value, the J_SMITH process is created on the VAX system with 250 as the maximum number of ASTs and scheduled wake-up requests.

Conclusion: Using AXP level process quotas in a common SYSUAF file might cause an overuse of resources by VAX processes in the VMScluster.

Scenario 2: VAX Level Values in Common SYSUAF, AXP Host Node

Assume that a common SYSUAF.DAT file is used in a dual-architecture VMScluster and that the cluster manager elects to use process quotas equal to the default values from a VAX SYSUAF.DAT file. As shown in Table 2–6, the default values are lower on VAX VMS Version 5.5–2 systems (as compared with OpenVMS AXP Version 1.5 systems). A user logs in to the VMScluster and the J_SMITH process is about to be created *on an AXP system.*

Assume that the ASTLM value of 24 is defined in the common SYSUAF.DAT record for J_SMITH. This value is the one used by default in VAX SYSUAF.DAT files. When the J_SMITH process is created on the AXP system, LOGINOUT compares the ASTLM process quota value (24) with the active, corresponding PQL_MASTLM system parameter value on the host AXP system. Although the default PQL_MASTLM value on AXP systems is 4, assume that, after running AUTOGEN, the current PQL_MASTLM value on the host AXP system is 120. Because LOGINOUT uses the larger value, the J_SMITH process is created on the AXP system with 120 as the maximum number of ASTs and scheduled wake-up requests.

Conclusion: Using VAX level process quotas in a common SYSUAF file, examine the dynamic PQL_M*quota* values on the AXP system to ensure that the AXP process quotas are not too low. If necessary, increase the appropriate PQL_M*quota* values on the AXP system in MODPARAMS.DAT.

Scenario 3: AXP Level Values in Common SYSUAF, AXP Host Node

Assume that a common SYSUAF.DAT file is used in a dual-architecture VMScluster and that the cluster manager chooses to use process quotas equal to the default values from an AXP SYSUAF.DAT file. As shown in Table 2–6, the default values are higher on OpenVMS AXP Version 1.5 systems (as compared with VAX VMS Version 5.5–2 systems). A user logs in to the VMScluster and the J_SMITH process is about to be created *on an AXP system.*

Assume that the ASTLM value of 250 is defined in the common SYSUAF.DAT record for J_SMITH. This value is the one used by default in AXP SYSUAF.DAT files. When the J_SMITH process is about to be created on the AXP system, LOGINOUT compares the ASTLM process quota value (250) with the active, corresponding PQL_MASTLM system parameter value on the host AXP system. Although the default PQL_MASTLM value on AXP systems is 4, assume that, after running AUTOGEN, the current PQL_MASTLM value on the host AXP system is 120. Because LOGINOUT uses the larger value, the J_SMITH process is created on the AXP system with 250 as the maximum number of ASTs and scheduled wake-up requests.

Conclusion: Using the AXP level process quotas in a common SYSUAF works fine for the AXP processes, but, as in Scenario 1, the AXP level process quotas in a common SYSUAF might cause overuse of resources by the VAX processes.

Scenario 4: VAX Level Values in Common SYSUAF, VAX Host Node

Assume that a common SYSUAF.DAT file is used in a dual-architecture VMScluster and that the cluster manager elects to use process quotas equal to the default values from a VAX SYSUAF.DAT file. As shown in Table 2–6, the default values are lower on VAX VMS Version 5.5–2 systems (as compared with OpenVMS AXP Version 1.5 systems). A user logs in to the VMScluster and the J_SMITH process is about to be created *on a VAX system.*

Assume that the ASTLM value of 24 is defined in the common SYSUAF.DAT record for J_SMITH. This value is the one used by default in VAX SYSUAF.DAT files. When the J_SMITH process is created on the VAX system, LOGINOUT compares the ASTLM process quota value (24) with the active, corresponding PQL_MASTLM system parameter value on the host VAX system. Although the default PQL_MASTLM value on VAX systems is 4, assume that after running AUTOGEN, the current PQL_MASTLM value on the host VAX system is 100. Because LOGINOUT uses the larger value, the J_SMITH process is created on the VAX system with 100 as the maximum number of ASTs and scheduled wake-up requests.

Conclusion: Using the VAX level process quotas in a common SYSUAF works fine for the VAX processes, but, as in Scenario 2, this might restrict the resources needed by the AXP processes. If necessary, you can increase the appropriate PQL_M*quota* values on the AXP system without disrupting the needs of the VAX processes in the VMScluster.

Conclusions

In a common SYSUAF.DAT file for a dual-architecture VMScluster, consider using the process quota values that seem most appropriate for your VAX processes. The VAX level SYSUAF values should:

- Continue to work well for the processes that are created on the VAX nodes in the dual-architecture VMScluster.
- Not interfere with the needs of processes that are created on the AXP nodes in the VMScluster. Using this approach, you should monitor the PQL_M*quota* parameters on the AXP system and, if necessary, adjust these values in MODPARAMS.DAT so that the AXP processes receive the needed resources.

Table 2–7 summarizes the common SYSUAF.DAT scenarios in a dual-architecture VMScluster and the probable results.

Table 2–7 Summary of Common SYSUAF.DAT Scenarios and Probable Results

Scenario	Probable Results
AXP level values in common SYSUAF in a dual-architecture VMScluster	A process that starts on an AXP node executes with the values you deem appropriate. For a process that starts on a VAX node, the likely result is that LOGINOUT will not use the system-specific PQL_M*quota* values defined on the VAX system because LOGINOUT finds higher values for each quota in the AXP style SYSUAF.DAT. This could cause inappropriately high resource use by VAX processes in the VMScluster.

(continued on next page)

Table 2–7 (Cont.) Summary of Common SYSUAF.DAT Scenarios and Probable Results

Scenario	Probable Results
VAX level values in common SYSUAF in dual-architecture VMScluster	A process that starts on a VAX node executes with the values you deem appropriate. For a process that starts on an AXP node, the likely result is that LOGINOUT will ignore the typically lower VAX level values in the SYSUAF and instead will use the value of each quota's current PQL_M*quota* value on the AXP system. Monitor the current values of PQL_M*quota* system parameters if you try this approach. Increase the appropriate PQL_M*quota* values on the AXP system in MODPARAMS.DAT as necessary.

Still, you might decide to experiment with the higher process quota values that usually are associated with an OpenVMS AXP system's SYSUAF.DAT, as you determine values for a common SYSUAF.DAT in a VMScluster environment. The higher AXP level process quotas might be appropriate for processes created on host VAX nodes in the VMScluster if the VAX systems have large, available memory resources. The values that are appropriate for processes on your VAX and AXP systems will be determined by experimentation and modification over time. Factors in your decisions about appropriate limit and quota values for each process will include the amount of available memory, CPU processing power, the average work load of the applications, and peak work loads of the applications.

See *VMScluster Systems for OpenVMS* for important details about setting up a common SYSUAF.DAT file in a dual-architecture VMScluster.

2.2.27 Impact of Translated Code on Disk Quotas and Virtual Memory Use

The process of using SYSMAN DISKQUOTA to create, add, enable, disable, modify, rebuild, remove, and show the disk quotas assigned to a system user is the same. You might need to increase disk quotas on AXP computers that store translated images from VAX computers.

Translated images might require more virtual memory than equivalent OpenVMS VAX images. Also, native OpenVMS AXP images (having been linked to the larger page size) consume more virtual memory.

Consequently, you might need a larger page file quota. The default values for related memory quotas have been adjusted as follows for OpenVMS AXP:

- As noted in Section 2.2.25, the PGFLQUOTA default value is increased from 10240 (512-byte pages) on VAX VMS Version 5.5–2 to 50000 (512-byte pagelets) on OpenVMS AXP.
- The VIRTUALPAGECNT system parameter default value is increased from 9216 (512-byte pages) on VAX VMS Version 5.5–2 to 65536 (512-byte pagelets) on OpenVMS AXP. VIRTUALPAGECNT is the maximum virtual page count. The parameter determines the total number of pages that can be mapped for a process, which can be divided in any fashion between P0 and P1 space.
- The PQL_DPGFLQUOTA system parameter default value is increased from 8192 (512-byte pages) on VAX VMS Version 5.5–2 to 65536 (512-byte pagelets) on OpenVMS AXP. PQL_DPGFLQUOTA is the default paging-file quota.

Refer to Section 5.1 and Section 5.2 for related information.

2.2.28 Terminal Fallback Facility

The OpenVMS Terminal Fallback Utility (TFU) is the user interface to the OpenVMS Terminal Fallback Facility (TFF). This facility provides table-driven character conversions for terminals. TFF includes a fallback driver (SYS$FBDRIVER.EXE), a shareable image (TFFSHR.EXE), a terminal fallback utility (TFU.EXE), and a fallback table library (TFF$MASTER.DAT).

- To start TFF, invoke the TFF startup command procedure located in SYS$MANAGER, as follows:

```
$ @SYS$MANAGER:TFF$SYSTARTUP.COM
```

- To enable fallback or to change fallback characteristics, invoke TFU as follows:

```
$ RUN SYS$SYSTEM:TFU
TFU>
```

- To enable default fallback to the terminal, issue the following DCL command:

```
$ SET TERMINAL/FALLBACK
```

The OpenVMS AXP TFF differs from the OpenVMS VAX TFF in the following ways:

- On OpenVMS AXP, the TFF fallback driver is SYS$FBDRIVER.EXE. On OpenVMS VAX, the TFF fallback driver is FBDRIVER.EXE.
- On OpenVMS AXP, the TFF startup file is TFF$*SY*STARTUP.COM. On OpenVMS VAX, the TFF startup file is TFF$STARTUP.COM.
- On OpenVMS AXP, TFF can handle 16-bit character fallback. The fallback table library (TFF$MASTER.DAT) contains two more 16-bit character tables than on OpenVMS VAX. These two tables are used mainly by the Asian region. Also, the table format was changed in order to support of 16-bit character fallback.
- On OpenVMS AXP, the TFU command SHOW STATISTICS does not display the size of the fallback driver (SYS$FBDRIVER.EXE).

TFF does not support RT terminals.

Refer to the *VMS Terminal Fallback Utility Manual* for more information about TFF.

3
Maintenance Tasks

Most OpenVMS AXP system management maintenance tasks are identical to those on OpenVMS VAX. This chapter:

- Identifies which OpenVMS system management maintenance tasks are the same on AXP and VAX
- Explains how some OpenVMS AXP system management maintenance tasks are different from those of OpenVMS VAX

3.1 Maintenance Tasks That Are the Same

Table 3–1 list the OpenVMS system management maintenance tasks that are identical or similar on AXP and VAX.

Table 3–1 Identical or Similar VMS Maintenance Tasks

Feature, Task, or Command	Comments
File system	All basic file system support is present. Note that Files–11 On-Disk Structure Level 1 (ODS-1) format disks and multivolume file sets are not supported on OpenVMS AXP.
ALLOCATE command	Identical.
MOUNT command	The same except for MOUNT/SHADOW, which is not supported because the OpenVMS volume shadowing feature is not available on OpenVMS AXP at this time.
Accounting utility commands	Identical.
Analyzing error logs	The ANALYZE/ERROR_LOG command is the same as on OpenVMS VAX systems, with one exception: the /SUMMARY qualifier is not supported. On OpenVMS AXP, an error results if you attempt to read an error log of a VAX computer.
ANALYZE/OBJECT and ANALYZE/IMAGE commands	Command format is identical on VAX and AXP systems. You or your system's programmers should plan ahead and manage the location of native VAX VMS .OBJ and .EXE files and the location of native OpenVMS AXP .OBJ and .EXE files.
ANALYZE/PROCESS_DUMP command	Format is identical on VAX and AXP.

(continued on next page)

Table 3–1 (Cont.) Identical or Similar VMS Maintenance Tasks

Feature, Task, or Command	Comments
Other ANALYZE commands, /AUDIT, /CRASH_DUMP, /DISK_STRUCTURE, /MEDIA, /RMS_FILE, /SYSTEM	Identical.
Backing up data	With one exception (/EXACT_ORDER qualifier added in OpenVMS VAX Version 6.0), the BACKUP command and qualifiers are the same on OpenVMS AXP. Note: Thoroughly read the *OpenVMS AXP Version 1.5 Release Notes* for the latest information about any restrictions with backup and restore operations on AXP and VAX systems. (See Section 3.2.2 for information about BACKUP /EXACT_ORDER.)
Batch and print queuing system	Maintenance of the OpenVMS AXP batch and print queuing system (creating, stopping and restarting queues) is the same as in VMS Version 5.5–2. However, the batch and print queuing system is much different since OpenVMS AXP Version 1.0, when batch and print was the same as in VMS Version 5.4. See Section 3.2.1 for a comparison of the batch and print queuing systems on recent releases of the operating system.
CONVERT, CONVERT/RECLAIM, and the CONVSHR shareable library	Identical.
MONITOR ALL_CLASSES command	The same, with exceptions. Does not include the NONPAGED POOL STATISTICS class because MONITOR POOL is not supported on OpenVMS AXP. See Section 3.2.4 and Section 5.4 for more information.
MONITOR MODES	The same, with one display exception: MONITOR MODES initiates monitoring of the TIME IN PROCESSOR MODES class, which includes a data item for each mode of processor operation. In displays, Interrupt Stack is replaced by Interrupt State because AXP computers do not have an interrupt stack, and service interrupts occur on the current process's kernel stack.
MONITOR/RECORD and MONITOR/INPUT	Identical. Also, MONITOR/INPUT on an AXP node can read a MONITOR.DAT file created by MONITOR/RECORD on a VAX node, and vice versa.
MONITOR TRANSACTION	Identical.
MONITOR VECTOR	Displays zeros for any AXP processor, where vectors are not supported. On an AXP computer, MONITOR VECTOR operates the same as on a VAX computer without vector processing.

(continued on next page)

Table 3–1 (Cont.) Identical or Similar VMS Maintenance Tasks

Feature, Task, or Command	Comments
Other MONITOR commands	The format for the following commands is unchanged: MONITOR CLUSTER, MONITOR DECNET, MONITOR DISK, MONITOR DLOCK, MONITOR FCP, MONITOR FILE_SYSTEM_CACHE, MONITOR IO, MONITOR LOCK, MONITOR PAGE, MONITOR PROCESSES, MONITOR RMS, MONITOR STATES, MONITOR SYSTEM.
SUMSLP	Identical.
SYSMAN utility	Similar utility functions; however, the I/O subsystem configuration functions from the OpenVMS VAX SYSGEN utility are now in the OpenVMS AXP SYSMAN utility. See Section 2.2.12 and Appendix B for details.
System Dump Analyzer (SDA)	All .STB files that are available to the System Dump Analyzer (SDA) on OpenVMS VAX are available on OpenVMS AXP systems. (Note: the .STB files are in SYS$LOADABLE_IMAGES and not in SYS$SYSTEM.) System dump-file size requirements are higher on OpenVMS AXP systems. See Section 3.2.3 for more information about SDA differences.

3.2 Maintenance Tasks That Are Different

This section describes the OpenVMS system management maintenance tasks that are different on AXP systems. The differences are:

- The batch and print queuing system for OpenVMS AXP Version 1.5 is the same as the enhanced, clusterwide batch and print queuing in VAX VMS Version 5.5. The OpenVMS AXP Version 1.5 batch and print queuing is much different from the VMS Version 5.4 batch and print system. The OpenVMS VAX Version 6.0 batch and print queuing system contains enhancements to the system seen in VMS Version 5.5 and OpenVMS AXP Version 1.5. See Section 3.2.1.
- The BACKUP command in OpenVMS VAX Version 6.0 adds the /EXACT_ORDER qualifier, which is not supported on OpenVMS AXP Version 1.5 at this time. See Section 3.2.2.
- Larger system dump files occur and new values for the DUMPSTYLE system parameter are provided. See Section 3.2.3.
- The MONITOR POOL command is not provided (also true on OpenVMS VAX Version 6.0 systems). See Section 3.2.4.
- The OpenVMS **movefile** subfunction is not supported. See Section 3.2.5.
- No Patch utility is supplied for OpenVMS AXP systems. See Section 3.2.6.

3.2.1 Comparison of Batch and Print Queuing Systems

Table 3–2 compares the batch and print queuing systems for recent releases of the operating system.

Table 3–2 Comparison of Batch and Print Queuing Systems

VMS Version 5.4	VMS Version 5.5 and OpenVMS AXP Version 1.5	OpenVMS VAX Version 6.0
Queue manager runs on each node in a cluster; no failover.	Clusterwide operation; queue manager failover to a surviving node.	Clusterwide operation, queue manager failover to a surviving node; option of multiple queue managers to distribute batch and print work load between VAXcluster nodes (to work around CPU or memory resource shortages.)
Queue manager runs as part of each node's job controller process.	Queue manager and job controller functions are separate.	Queue manager and job controller functions are separate.
Shared queue database, JBCSYSQUE.DAT.	Centralized queue database: QMAN$MASTER.DAT (master file); SYS$QUEUE_MANAGER.QMAN$QUEUES (queue file), and SYS$QUEUE_MANAGER.QMAN$JOURNAL (journal file).	Same centralized queue database files as in VMS Version 5.5 and OpenVMS AXP Version 1.5; for each additional queue manager, the queue database contains a queue file and journal file; format is *name-of-manager*.QMAN$QUEUES and *name-of-manager-*.QMAN$JOURNAL.
START/QUEUE/MANAGER command.	START/QUEUE/MANAGER command has /ON=(*node-list*) qualifier to specify order in which nodes claim the queue manager during failover.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5, but also has /ADD and /NAME_OF_MANAGER=*queue-manager-name* to create additional queue managers and distribute work load of print and queue functions in the VAXcluster.
START/QUEUE/MANAGER command has /EXTEND, /BUFFER_COUNT, /RESTART qualifiers.	Obsolete.	Obsolete.
No autostart.	Autostart feature lets you start autostart queues on a node with a single command; also lets you specify a list of nodes in the VMScluster to which a queue can fail over automatically.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5.

(continued on next page)

Table 3–2 (Cont.) Comparison of Batch and Print Queuing Systems

VMS Version 5.4	VMS Version 5.5 and OpenVMS AXP Version 1.5	OpenVMS VAX Version 6.0
INITIALIZE /QUEUE command and START /QUEUE command.	New or changed commands with autostart feature: • INITIALIZE /QUEUE / AUTOSTART_ ON=[(*node*::[*device*] [,...]) • ENABLE AUTOSTART [/QUEUES] [/ON_ NODE=*node-name*] • START /QUEUE / AUTOSTART_ ON=[(*node*::[*device*] [,...]) • DISABLE AUTOSTART [/QUEUES] [/ON_ NODE=*node-name*]	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5; however, ENABLE AUTOSTART and DISABLE AUTOSTART also include the new /NAME_OF_MANAGER qualifier.
/RETAIN qualifier can be used with INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command.	/RETAIN qualifier can also be used with PRINT, SUBMIT, or SET ENTRY command.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5.
SHOW ENTRY command display lists job name, user name, entry number, blocks, status, and name of queue.	SHOW ENTRY command display format is slightly different, making it easier to find a job's entry number. Also, SHOW ENTRY accepts job names to narrow the display criteria.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5.
SHOW QUEUE command display lists job name, user name, entry number, status, name of queue, and node on which job is running.	SHOW QUEUE command display format is slightly different, making it easier to find a job's entry number.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5, but also adds /MANAGER qualifier to display information about one or more queue managers.
SUBMIT command.	SUBMIT command adds /NOTE qualifier, which lets you specify a message of up to 255 characters.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5.
F$GETQUI lexical function.	F$GETQUI lexical function enhanced to return information about the new autostart feature.	Same as in VMS Version 5.5 and OpenVMS AXP Version 1.5, but also adds information about the new manager-specific features.
$GETQUI and $SNDJBC system services.	$GETQUI and $SNDJBC system services enhanced to support new batch and print queuing system.	Further enhanced due to multiple queue managers; includes new parameters.

(continued on next page)

Table 3–2 (Cont.) Comparison of Batch and Print Queuing Systems

VMS Version 5.4	VMS Version 5.5 and OpenVMS AXP Version 1.5	OpenVMS VAX Version 6.0
UIC-based protection of queues; default access is System:Execute, Owner:Delete, Group:Read, and World:Write.	Same as in VMS Version 5.4.	C2 security adds: • SHOW SECURITY /CLASS=QUEUE *queue-name* • SET SECURITY /CLASS=QUEUE /ACL=(ID=*uic*, ACCESS=*access*) *queue-name* • UIC-based protection of queues; default access is System:Manage, Owner:Delete, Group:Read, and World:Submit.

For details about the new batch and print queuing systems, refer to:

- *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*
- *OpenVMS DCL Dictionary*

3.2.2 BACKUP /EXACT_ORDER Not Supported

OpenVMS VAX Version 6.0 includes the /EXACT_ORDER qualifier to the BACKUP command. This new qualifier is not supported at this time on OpenVMS AXP.

Depending on the other qualifiers you specify on the BACKUP command line, /EXACT_ORDER on OpenVMS VAX Version 6.0 systems lets you:

- Specify the exact order of tapes and labels that you want to use in a BACKUP operation. You must use the /LABEL=(*label1*,*label2*,...) qualifier to specify the order of the labels. BACKUP continues the operation as long as the label of the tape in the drive matches the corresponding label on the command line. If you do not specify enough labels on the command line to complete the operation, BACKUP prompts you to enter a label for the tape in the drive.
- Preserve the existing volume label on a tape. If you do not use the /LABEL qualifier on the command line and the tape has an ANSI label, BACKUP uses the existing label.
- Prevent previous volumes of a multivolume save operation from being overwritten. BACKUP keeps track of the volume labels you have already used in the operation. If you accidentally mount one of the previous volumes, BACKUP displays the following error message:

```
%BACKUP-W-MOUNTERR, volume 1 on MKB100: was not mounted because its
label does not match the one requested Volume with label TAPE1 was
already used in this save operation specify option (QUIT or NEW tape)
```

3.2.3 System Dump Analyzer

Differences in the System Dump Analyzer (SDA) occur with the size of the system dump file. See Section 3.2.3.1 for more information. Also, see Section 3.2.3.2 for a related discussion about conserving dump file space.

See the *OpenVMS AXP System Dump Analyzer Utility Manual* and the *OpenVMS VAX System Dump Analyzer Utility Manual* for details about SDA.

3.2.3.1 Size of the System Dump File

The location and the file name of the system dump file is the same. However, VAX and AXP system dump-file size requirements differ. The following calculations apply to physical memory dump files.

On VAX systems, use the following formula to calculate the correct size for SYS$SYSTEM:SYSDUMP.DMP:

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

On AXP systems, use the following formula to calculate the correct size:

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

3.2.3.2 Conserving Dump File Storage Space

Dump file storage space might be at a premium on OpenVMS AXP computers. For system configurations with large amounts of memory, the system dump files that are automatically created can be extremely large. For example, on a 256MB system, AUTOGEN creates a dump file in excess of 500,000 blocks.

One way to conserve dump file storage space is to provide selective dumps rather than full dumps. This is *vital* on very large memory systems.

Use the DUMPSTYLE system parameter to set the desired method for capturing system dumps on BUGCHECK. On OpenVMS VAX systems, the parameter can be set to one of two values. DUMPSTYLE can be set to one of four values on OpenVMS AXP Version 1.5. When a system fails on a VAX or AXP computer, a lot of data is printed to the operator's console (if one exists); when this step completes, only then are the memory contents written fully or selectively to the dump file. Some VAX and AXP computers might not have consoles, in which case this console data never appears.

VAX systems always have full console output. On AXP systems, the information is more complex and full console output is much longer (although it contains the same basic information). The DUMPSTYLE system parameter for OpenVMS AXP Version 1.5 introduces options for shorter versions of the console output. Digital picked the values 0 and 1 for the shorter console output on OpenVMS AXP systems so that you do not have to change your DUMPSTYLE system parameter to get the default, shorter output.

Table 3–3 compares the values for the DUMPSTYLE parameter.

Table 3–3 Comparison of DUMPSTYLE System Parameter Values

Value	Meaning on OpenVMS VAX	Meaning on OpenVMS AXP Version 1.5
0	Full console output; full dump	Minimal console output; full dump
1	Full console output; selective dump	Minimal console output; selective dump
2	Does not exist	Full console output; full dump
3	Does not exist	Full console output; selective dump

On AXP and VAX systems, a SHOW command in the SYSGEN utility lists the default value for the DUMPSTYLE system parameter as 0. However, on OpenVMS AXP, the AUTOGEN calculated value (effectively a default) is 1.

You can use the following SYSGEN commands to modify the system dump file size on large memory systems:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CREATE SYS$SYSTEM:SYSDUMP.DMP/SIZE=70000
$ @SHUTDOWN
```

After the system reboots (and *only* after a reboot), you can purge SYSDUMP.DMP.

The dump file size of 70,000 blocks is sufficient to cover about 32MB of memory. This dump file size almost always encompasses the information needed to analyze a system failure.

3.2.4 MONITOR POOL Command Not Provided

The MONITOR POOL command, which on VMS Version 5.5 and earlier systems initiates monitoring of the NONPAGED POOL STATISTICS class and measures space allocations in the nonpaged dynamic pool, is not provided on OpenVMS AXP or on OpenVMS VAX Version 6.0 systems. This is due to adaptive pool management features and two SDA commands on OpenVMS AXP: SHOW POOL/RING_BUFFER and SHOW POOL/STATISTICS. See Section 5.4 for more information.

3.2.5 Defragmenting Disks

The OpenVMS **movefile** subfunction, which lets programmers write atomic-file disk defragmentation applications, was introduced with OpenVMS VAX Version 5.5. The **movefile** subfunction is not supported on OpenVMS AXP. The DEC File Optimizer for VMS layered product, which defragments disks using the **movefile** subfunction, also is not supported on OpenVMS AXP.

3.2.6 Patch Utility Not Supported

The OpenVMS VAX Patch utility is not supported on OpenVMS AXP because compiler optimizations and the AXP architecture make the placement of instructions and data in an image much more complex.

The *OpenVMS Calling Standard* defines a component of each module known as a **linkage section**. You cannot make calling standard–conformant calls to routine entry points outside of the current module (or access another module's data) without referencing the linkage section. Thus, you cannot patch image code without also patching the appropriate linkage section. Patching a linkage section is difficult if you do not know what the compiler and linker have done to establish the linkage section as it appears to the image activator. For those reasons, a patch utility is not available on OpenVMS AXP.

4 Security Tasks

Security tasks are those you perform to ensure the protection and integrity of applications and data on your OpenVMS AXP system. Security tasks on OpenVMS AXP Version 1.5 are the same as on VAX VMS Version 5.5 and earlier releases. This chapter focuses on the differences with OpenVMS VAX Version 6.0, which supports C2 security features.

See the *OpenVMS AXP Guide to System Security* for details about the OpenVMS AXP security features. The *OpenVMS VAX Guide to System Security* decribes the C2 security features that are summarized in this chapter.

4.1 Overview of C2 Security Features on OpenVMS VAX Version 6.0

OpenVMS VAX Version 6.0 offers significant enhancements to system security in the areas of object protection and auditing.

OpenVMS VAX Version 6.0 protects more classes of objects, and each one of these classes carries a consistent set of security elements. The system now protects more object classes, including volume, common event flag cluster, resource domain, and security class. Earlier versions of VMS, as well as OpenVMS AXP Version 1.5, support access control lists (ACLs) on a number of objects, such as files and queues. With OpenVMS VAX Version 6.0, ACLs can be assigned to any class of object. Thus, every object carries a protection code, an ACL, and a user identification code (UIC). OpenVMS VAX Version 6.0 allows you to change these elements after the objects are created and provides a new interface to simplify management of object protection.

OpenVMS VAX Version 6.0 extends support for the auditing system for reporting the use of protected objects. In conjunction with these security changes, the following enhancements are also provided:

- Modification of the rights database (SYS$SYSTEM:RIGHTSLIST.DAT) to deny world read access
- More sophisticated access control through protected subsystems
- Support for running OpenVMS VAX in a C2 environment (see Section 4.5)
- Expanded use of rights identifiers
- Greater control of files created within directories that are owned by a resource identifier
- Added privileges for supporting SEVMS, a security-enhanced version of the OpenVMS operating system

For detailed information about any of the changes described in the following sections, see the *OpenVMS VAX Guide to System Security.*

4.2 Object Protection

This section discusses object protection on systems running OpenVMS VAX Version 6.0.

4.2.1 Consistent Set of Security Elements for All Object Classes

With OpenVMS VAX Version 6.0, all protected objects carry a common set of elements in a **security profile**. A security profile includes an access control list (ACL), a UIC-based protection code, and an owner UIC. Previous versions of VMS, as well as OpenVMS AXP Version 1.5, support full security profiles for the following classes of objects: files, devices, global sections, logical name tables, queues, and capabilities.

4.2.2 New Classes of Objects

OpenVMS VAX Version 6.0 protects the following object classes:

- Common event flag clusters—A set of 32 event flags that enable cooperating processes to post event notifications to each other.
- Volumes—Single point of access control to all files contained on a Files–11 On-Disk Structure Level 2 (ODS-2) disk volume.
- Resource domains—A namespace controlling access to lock management resources. OpenVMS VAX Version 6.0 protects access to locks (and their associated lock value blocks) by protecting the resource domain to which the lock resource belongs.
- Security classes—A new object class that includes all other object classes as its members. This class defines the template profiles for each object class (excluding files). Template profiles provide protection, ownership, and ACL attributes for newly created objects. (Thus, system parameters TTY_OWNER and TTY_PROT are obsolete.)

4.2.3 New Security Interfaces

On systems running OpenVMS VAX Version 6.0, you can manage all protected objects through the new DCL commands SET SECURITY and SHOW SECURITY or the new $SET_SECURITY and $GET_SECURITY system services.

The SET SECURITY command modifies the security profile of an object (see Section 4.2.1). The SHOW SECURITY command displays the name, class, and profile of a protected object.

4.2.4 Security Profile Templates

OpenVMS VAX Version 6.0 supplies default templates for all classes of objects other than files. When you create an object, the system uses the appropriate template to construct an initial security profile for the object. You can modify any profile template to change the default values by using the DCL command SET SECURITY. For example, you can specify a default ACL for newly created mailboxes.

The security elements of files are derived in the same way as in earlier versions of VMS (as well as OpenVMS AXP Version 1.5); for example, by using existing versions and directory defaults.

4.2.5 Appropriate Access Types for Each Object Class

Each object class defines access types appropriate to that class of object. For example, files support read, write, execute, and delete access, whereas queues support read, submit, manage, and delete access. Table 4–1 defines the access types for the specified object classes.

All objects support control access, which is the ability to examine or modify the elements within the security profile, that is, the protection code, ACL, and owner elements of an object.

Table 4–1 Access Types for Object Classes

Object Class	Access Types
Capability	Use, control
Common Event Flag Cluster	Associate, delete, control
Device	Read, write, physical, logical, control
File (including directory file)	Read, write, execute, delete, control
Group Global Section	Read, write, execute, control
Logical Name Table	Read, write, create, delete, control
Queue	Read, submit, manage, delete, control
Resource Domain	Read, write, lock, control
Security Class	Read, write, control
System Global Section	Read, write, execute, control
Volume	Read, write, create, delete, control

4.2.6 Device Protection

OpenVMS VAX Version 6.0 provides enhanced device protection. When mounting or initializing a volume, the operating system now checks access to the device as well as access to the volume.

To initialize or mount a volume on a device, you must have read, write, or control access to the device. System managers and users with the SYSPRV privilege have this type of access.

4.2.7 Clusterwide Distribution of Security Profiles

In OpenVMS AXP Version 1.5 and in VMS releases prior to Version 6.0, files were the only object class for which security elements, or profiles, were distributed throughout the cluster. With OpenVMS VAX Version 6.0, the operating system distributes security profiles throughout the cluster for all objects that are visible from multiple nodes in a cluster (files, disks, tapes, resource domains, and security classes). Thus, if an object is accessed by two processes in a cluster on different nodes of the cluster, the system uses the same security profile for both access checks. If you change the security profile for a cluster-visible object, the system automatically distributes your changes to the other nodes in the cluster.

A new security object database stores the cluster-visible profiles when the system shuts down. This database also maintains settings during a cluster reboot. Each time the system starts up, it is able to retrieve object profiles from the database. This retrieval process simplifies system startup files and helps you maintain a consistent security profile across the entire cluster.

4.3 Security Auditing

This section discusses security auditing on systems running OpenVMS VAX Version 6.0.

4.3.1 Distinction Between Alarm Events and Audit Events

In OpenVMS AXP Version 1.5 and in VMS releases prior to Version 6.0, all security-related events were reported as both alarm events and audit events. Alarm events go to security operator terminals only, whereas audits go to the audit log file. With OpenVMS VAX Version 6.0, these events can be enabled independently as alarm events, audit events, or both.

Note

The security audit log file, which resides in SYS$COMMON:[SYSMGR], is called SECURITY.AUDIT$JOURNAL on OpenVMS VAX Version 6.0. The file is called SECURITY_AUDIT.AUDIT$JOURNAL on OpenVMS AXP Version 1.5 and on VMS Version 5.5 and earlier releases.

4.3.2 Auditing for Additional Security Events

OpenVMS VAX Version 6.0 extends auditing to report on the following categories of events:

- Creation, access, deaccess, and deletion of objects
- Use (or failed use) of privilege
- Logical link connections or terminations through DECnet, DECwindows, or SYSMAN
- Changes to system parameters
- Use of identifiers as privilege
- Changes to the network configuration database, using the Network Control Program (NCP) utility
- Use of any process control system service
- Changes to the authorization databases
- Changes to system time

These extensions enable you to record more security-related events as they occur. You can enable auditing (which includes alarms) for these events by using the DCL command SET AUDIT. For more information, refer to the *OpenVMS VAX Guide to System Security*.

4.3.3 System Services That Support Auditing

The $AUDIT_EVENT and $CHECK_PRIVILEGE system services allow applications to report security-related events.

The $AUDIT_EVENT system service is called by the operating system any time a user attempts to perform a privileged function. The $CHECK_PRIVILEGE system service reports the use of privileges or identifiers to the $AUDIT_EVENT system service, which in turn reports the event to the audit server.

More information on the use of these system services for auditing is available in the *OpenVMS System Services Reference Manual*.

4.3.4 New Audit Access Control Entry

System managers can gain information on the use of individual objects. To enable alarms or audits for a specific object, system managers can use an Alarm access control entry (ACE) (this alarm function was formerly called ALARM_JOURNAL) or an Audit ACE, respectively.

4.3.5 New Mechanism for Controlling Audit Messages

OpenVMS VAX Version 6.0 limits the ability of processes to overload the system with audit messages. By restricting the number of messages, the system prevents excessive resource consumption and loss of critical audit records. The auditing subsystem monitors the total number of messages in the audit server's memory and tracks the number of messages contributed by each active process. At certain thresholds, the influx of messages is controlled. Message volume is controlled on a per-process basis.

4.4 Protected Subsystems

With OpenVMS VAX Version 6.0, you can use protected subsystems to provide more sophisticated access control on objects. A protected subsystem is an application which, when run, causes the process running the application to be granted one or more additional identifiers. For as long as a user runs the subsystem, the user's security profile carries these additional identifiers.

Users with execute access to the application can gain access to the subsystem. Once in the subsystem, users can work with the data files and other resources of the subsystem.

To support protected subsystems, OpenVMS VAX Version 6.0 includes the following features:

- A subsystem access control entry (ACE)
- A new Subsystem attribute for the AUTHORIZE commands GRANT/ID and REVOKE/ID, and the DCL command SET RIGHTS
- The new $SUBSYSTEM system service
- The new DCL command qualifier /SUBSYSTEM for the MOUNT and SET VOLUME commands
- The PRC$M_SUBSYSTEM flag for the $CREPRC routine and the SUBSYSTEM flag for the LIB$SPAWN routine

4.5 Running OpenVMS VAX in a C2 Environment

The National Computer Security Center (NCSC) has evaluated OpenVMS VAX Version 6.0 relative to federal standards for operating system security (DOD 5200.28-STD). The OpenVMS operating system meets the criteria of a Division C, class 2 system.

The *OpenVMS VAX Guide to System Security* describes the requirements for creating and maintaining a C2 environment. The rules for creating and maintaining a C2 environment are based on the *Trusted Facility Manual* and the *Security Features User's Guide.*

4.6 Additional Security Enhancements

This section discusses additional security enhancements on systems running OpenVMS VAX Version 6.0.

4.6.1 Facility-Specific Rights Identifiers

OpenVMS VAX Version 6.0 supports facility-specific rights identifiers. This type of rights identifier allows a system manager to give users rights without assigning user privileges.

During the product's installation, facility identifiers combine an application's facility code with the value specified by that application. The application then uses the $ADD_IDENT service to register the application's facility identifiers with the operating system.

For more information about the use of facility-specific rights identifiers, refer to the *OpenVMS Programming Concepts Manual.*

4.6.2 Rights Identifiers Attributes

OpenVMS VAX Version 6.0 also extends the use of rights identifiers with the following identifier attributes:

Holder Hidden	Prevents someone from getting a list of users who hold an identifier unless they own the identifier themselves.
Name Hidden	Allows holders of an identifier to have it translated (either from binary to ASCII or from ASCII to binary) but prevents unauthorized users from translating the identifier.
No Access	Makes any access rights of the identifier null and void. This attribute is intended as a modifier for a resource identifier or the subsystem attribute.
Subsystem	Allows holders of the identifier to create and maintain protected subsystems by assigning the Subsystem ACE to the application images in the subsystem.

A complete list of all identifier attributes is in the *OpenVMS VAX Guide to System Security.*

4.6.3 Files Created in a Resource Identifier's Directory

A process without system privileges can create a file in a directory owned by a resource identifier. In OpenVMS AXP Version 1.5 and in VAX VMS releases prior to OpenVMS VAX Version 6.0, a file created under these conditions was assigned an extra ACE with a combination of control and ownership access. The purpose of the extra ACE was to guarantee access to the file by its creator. However, depending on the explicit security controls set up in the directory, the extra ACE can interfere with the accessibility of the files. A new Creator ACE provides the tool to solve these problems.

In OpenVMS VAX Version 6.0, you can assign a Creator ACE to the parent directory to determine the access specified by the extra ACE. You can also use a Creator ACE to suppress the extra ACE. If there is a Creator ACE in the ACL for the parent directory, the system propagates the access specified in the Creator ACE to the new ACE. If a directory lacks a Creator ACE, the system adds an extra ACE, which gives you control access plus the access specified in the owner field of the file's protection code. A Creator ACE with ACCESS=NONE suppresses the addition of the extra ACE.

4.6.4 New Privileges

Four new privileges have been added to OpenVMS VAX Version 6.0: AUDIT, IMPORT, UPGRADE, and DOWNGRADE.

The AUDIT privilege allows software programs to append audit records to the system security audit log file using one of four system services: $AUDIT_EVENT, $CHECK_PRIVILEGE, $CHKPRO, or $CHECK_ACCESS. In addition, the $AUDIT_EVENT system service allows for all components of an audit message to be specified. As a result, this privilege permits the logging of events that appear to have come from the operating system or another user process.

The three other new privileges, IMPORT, UPGRADE, and DOWNGRADE, are specific to SEVMS, a security-enhanced version of the OpenVMS operating system.

For more information on these four new privileges, refer to the *OpenVMS VAX Guide to System Security*.

5 Performance Optimization Tasks

This chapter describes the OpenVMS system management performance optimization tasks that are different on AXP systems. The differences are in the following areas:

- Impact of the larger AXP page size on system parameter units. See Section 5.1.
- Changes to the default values for a number of system parameters. See Section 5.2.
- Use of page or pagelet values by OpenVMS utilities and DCL commands. See Section 5.3.
- Adaptive pool management feature. See Section 5.4.
- Installation of suitably linked main images and shareable images in a granularity hint region for improved performance. See Section 5.5.
- A new feature called the virtual I/O cache (also part of OpenVMS VAX Version 6.0) that reduces bottlenecks and improves performance. See Section 5.6.

Note

Refer to Appendix A for a discussion about why and how the OpenVMS AXP system characteristics have an impact on tuning considerations.

5.1 System Parameters: Measurement Change for Larger Page Size

As discussed in Section 1.1.1 and as illustrated in Figure 1–1, a VAX page is 512 bytes, and an AXP page can be 8KB, 16KB, 32KB, or 64KB. The initial set of AXP computers use a page size of 8KB (8192 bytes).

The larger page size for an AXP system requires a fresh look at some of the system parameters that are measured in units of VAX pages on OpenVMS VAX. The same 512-byte quantity called a page on VAX is called a pagelet on OpenVMS AXP.

The OpenVMS VAX term *page* is ambiguous in the following ways for certain system parameters in the AXP context:

- On OpenVMS VAX, “page” sometimes is used instead of disk block.
- On OpenVMS VAX, “page” sometimes is used to express a total byte size.
- On OpenVMS VAX, “page” sometimes represents a discrete memory page, regardless of the number of bytes within the page.

Certain constraints affect how some system parameters are evaluated by the operating system. For instance, the working set control parameters can be expressed in the $CREPRC system service. As a result, a strict interpretation of pagelet must be maintained for OpenVMS AXP users.

The system parameters and units affected by this ambiguity fall into the following categories on OpenVMS AXP:

- Units that have changed in name only. See Section 5.1.1.
- Units that are CPU specific. See Section 5.1.2.
- Parameters that have dual values (both external and internal values). See Section 5.1.3.

5.1.1 System Parameter Units That Changed in Name Only

Table 5–1 shows the system parameters whose units have changed in name only, from "page" on OpenVMS VAX to the new, more appropriate name on OpenVMS AXP.

Table 5–1 System Parameter Units That Changed in Name Only

Parameter	Unit
ACP_DINDXCACHE	Blocks
ACP_DIRCACHE	Blocks
ACP_HDRCACHE	Blocks
ACP_MAPCACHE	Blocks
ACP_WORKSET	Pagelets
CLISYMTBL	Pagelets
CTLIMGLIM	Pagelets
CTLPAGES	Pagelets
ERLBUFFERPAGES	Pagelets
IMGIOCNT	Pagelets
PIOPAGES	Pagelets
MINWSCNT	Pure number
TBSKIPWSL	Pure number

5.1.2 CPU-Specific System Parameter Units

Table 5–2 shows the units that remain as CPU-specific pages (8KB on the initial set of AXP computers).

Table 5–2 CPU-Specific System Parameter Units

Parameter	Unit
BORROWLIM	Pages
FREEGOAL	Pages (also made DYNAMIC)
FREELIM	Pages
GROWLIM	Pages
MPW_HILIMIT	Pages
MPW_LOLIMIT	Pages
MPW_LOWAITLIMIT	Pages
MPW_THRESH	Pages
MPW_WAITLIMIT	Pages
MPW_WRTCLUSTER	Pages
GBLPAGFIL	Pages
RSRVPAGCNT	Pages

5.1.3 System Parameters with Dual Values

Table 5–3 shows the parameter units that have dual values. The parameter units in this category have both an external value and an internal value on OpenVMS AXP.

Table 5–3 System Parameters with Dual Values

Parameter	External Unit	Internal Unit	Function
PAGTBLPFC	Pagelets	Pages	Default page table page-fault cluster size. Specifies the maximum number of page tables to attempt to read to satisfy a fault for a nonresident page table.
PFCDEFAULT	Pagelets	Pages	Default page fault cluster size. During execution of programs on AXP systems, controls the number of image pagelets (pages, internally) read from disk per I/O operation when a page fault occurs. The value should not be greater than one-fourth the default size of the average working set to prevent a single page fault from displacing a major portion of a working set. Too large a value for PFCDEFAULT can hurt system performance. PFCDEFAULT can be overridden on an image-by-image basis with the CLUSTER option of the OpenVMS linker.
SYSPFC	Pagelets	Pages	Page fault cluster for system paging. The number of pagelets (pages, internally) read from disk on each system paging operation.
GBLPAGES	Pagelets	Pages	Global page table entry (PTE) count. Establishes the size in pagelets (pages, internally) of the global page table and the limit for the total number of global pages that can be created.

(continued on next page)

Table 5–3 (Cont.) System Parameters with Dual Values

Parameter	External Unit	Internal Unit	Function
SYSMWCNT	Pagelets	Pages	System working set count. Establishes the number of pagelets (pages, internally) for the working set containing the currently resident pages of pageable system space.
WSMAX	Pagelets	Pages	Maximum size of process working set. Determines the systemwide maximum size of a process working set, regardless of process quota.
VIRTUALPAGECNT	Pagelets	Pages	Maximum virtual page count. Determines the total number of pagelets (pages, internally) that can be mapped for a process, which can be divided in any fashion between P0 and P1 space.
WSINC	Pagelets	Pages	Working set increment. Sets the size in pagelets (pages, internally) to increase the working set size to compensate for a high page-fault rate.
WSDEC	Pagelets	Pages	Working set decrement. Sets the number of pagelets (pages, internally) to decrease the working set to compensate for a page-fault rate below the lower threshold.
AWSMIN	Pagelets	Pages	Establishes the lowest number of pagelets (pages, internally) to which a working set limit can be decreased by automatic working set adjustment.
SWPOUTPGCNT	Pagelets	Pages	Desired process page count for an outswap swap. This parameter sets the number of pagelets (pages, internally) to attempt to reduce a working set to before starting the outswap.
PQL_DPGFLQUOTA	Pagelets	Pages	Default paging file quota.
PQL_MPGFLQUOTA	Pagelets	Pages	Minimum paging file quota.
PQL_DWSDEFAULT	Pagelets	Pages	Default working set default size.
PQL_MWSDEFAULT	Pagelets	Pages	Minimum working set default size.
PQL_DWSQUOTA	Pagelets	Pages	Default working set quota.
PQL_MWSQUOTA	Pagelets	Pages	Minimum working set quota.
PQL_DWSEXTENT	Pagelets	Pages	Default working set extent.
PQL_MWSEXTENT	Pagelets	Pages	Minimum working set extent.

The external value is expressed in pagelets, and is accepted as input in $CREPRC or returned by the $GETSYI system service. Both SYSGEN and conversational bootstrap display both the internal and external parameter values. For example, the following is an edited SYSGEN display on OpenVMS AXP:

```
Parameter Name          Current   Default     Min.     Max.    Unit  Dynamic
--------------          -------   -------  -------  -------    ----  -------
PFCDEFAULT                  128       128        0     2032 Pagelets    D
 internal value               8         8        0      127 Pages       D

GBLPAGES                  81000     20000    10240       -1 Pagelets
 internal value            5063      1250      640       -1 Pages
SYSMWCNT                   2010      2000      512    16384 Pagelets
 internal value             126       125       32     1024 Pages
WSMAX                     65500      4000     2048   800000 Pagelets
 internal value            4094       250      128    50000 Pages
VIRTUALPAGECNT           139072     65536     2048  1000000 Pagelets
 internal value            8692      4096      128    62500 Pages
WSINC                      1200      2400        0       -1 Pagelets    D
 internal value              75       150        0       -1 Pages       D
WSDEC                       250       250        0       -1 Pagelets    D
 internal value              16        16        0       -1 Pages       D
AWSMIN                     2000      2000        0       -1 Pagelets    D
 internal value             125       125        0       -1 Pages       D
SWPOUTPGCNT                 512       512        0       -1 Pagelets    D
 internal value              32        32        0       -1 Pages       D
PQL_DPGFLQUOTA            65536     65536       -1       -1 Pagelets    D
 internal value            4096      4096     4096       -1 Pages       D
PQL_MPGFLQUOTA            65536      2048       -1       -1 Pagelets    D
 internal value            4096       128      128       -1 Pages       D
PQL_DWSDEFAULT             2000      2000       -1       -1 Pagelets
 internal value             125       125      125       -1 Pages
PQL_MWSDEFAULT             2000      2000       -1       -1 Pagelets
 internal value             125       125      125       -1 Pages
PQL_DWSQUOTA               6000      4000       -1       -1 Pagelets    D
 internal value             375       250      250       -1 Pages       D
PQL_MWSQUOTA               4000      4000       -1       -1 Pagelets    D
 internal value             250       250      250       -1 Pages       D
PQL_DWSEXTENT             65500     12000       -1       -1 Pagelets    D
 internal value            4094       750      750       -1 Pages       D
PQL_MWSEXTENT              6000      4000       -1       -1 Pagelets    D
 internal value             375       250      250       -1 Pages       D
```

Notice how the system parameter external default values (those in units of pagelets) are always multiples of 16 on an AXP system with 8KB pages. When a user specifies a pagelet value, that value is rounded up internally (if necessary) to the next whole page count because the operating system uses them in units of whole pages only, where each 8KB AXP memory page consists of 16 pagelets.

This characteristic has an important effect on system tuning. For example, you can increase a given parameter's external value by a single pagelet but not observe any effect on the behavior of the system. Because each AXP memory page consists of 16 pagelets, the parameter must be adjusted in multiples of 16 in order to change the internal value used by the operating system.

Refer to Section 2.2.25 for a related discussion of the rounding-up process with process quotas. Also, see Figure 1–1 for an illustration of the relative sizes of a VAX page, an AXP pagelet, and an AXP 8KB page.

5.2 Comparison of System Parameter Default Values

Table 5–4 shows the OpenVMS AXP system parameters whose default values, as noted in SYSGEN, are different from the value on OpenVMS VAX.

Note

Table 5–4 does not repeat the OpenVMS AXP system parameters listed in Table 5–3 when the only difference is in the name of the unit (a 512-byte VAX page to a 512-byte AXP pagelet). For example, the PFCDEFAULT default value on a VAX system is 32 pages; its default value on an AXP system is 32 pagelets, the same quantity.

Also, as you compare the columns, remember:

- Each AXP pagelet is the same quantity as each VAX page (512 bytes).
- Each CPU-specific AXP page (8192 bytes per page on AXP computers with 8KB pages) is 16 times larger than each VAX page (512 bytes per page).

Figure 1–1 illustrates the relative sizes of a VAX page, an AXP pagelet, and an AXP 8KB page.

Table 5–4 Comparison of System Parameter Default Values

Parameter	VAX Value	AXP Value
GBLPAGES	10000 pages	20000 pagelets
GBLPAGFIL	1024 pages	128 pages[1]
SYSMWCNT	500 pages	2000 pagelets
BALSETCNT	16 slots	32 slots
WSMAX	1024 pages	4000 pagelets
NPAGEDYN	430080 bytes	430000 bytes
PAGEDYN	210004 bytes	210000 bytes
VIRTUALPAGECNT	9216 pages	65536 pagelets
SPTREQ	3900 pages	Obsolete
MPW_WRTCLUSTER	120 pages	64 pages
MPW_HILIMIT	500 pages	512 pages
MPW_LOLIMIT	32 pages	16 pages
MPW_THRESH	200 pages	16 pages
MPW_WAITLIMIT	620 pages	576 pages
MPW_LOWAITLIMIT	380 pages	448 pages
AWSMIN	50 pages	2000 pagelets
SWPOUTPGCNT	288 pages	512 pagelets
MAXBUF	2064 bytes	8192 bytes
CLISYMTBL	250 pages	500 pagelets
LNM*S*HASHTBL	128 entries	512 entries
LNM*P*HASHTBL	128 entries	512 entries
PQL_DBIOLM	18 I/Os	32 I/Os

[1] Notice that 128 AXP pages (8192 bytes per page) are twice as large as 1024 VAX pages (512 bytes per page).

(continued on next page)

Table 5–4 (Cont.) Comparison of System Parameter Default Values

Parameter	VAX Value	AXP Value
PQL_DBYTLM	8192 bytes	65536 bytes
PQL_DDIOLM	18 I/Os	32 I/Os
PQL_DFILLM	16 files	128 files
PQL_DPGFLQUOTA	8192 pages	65536 pagelets
PQL_MPGFLQUOTA	512 pages	2048 pagelets
PQL_DPRCLM	8 processes	32 processes
PQL_DTQELM	8 timers	16 timers
PQL_DWSDEFAULT	100 pages	2000 pagelets
PQL_MWSDEFAULT	60 pages	2000 pagelets
PQL_DWSQUOTA	200 pages	4000 pagelets
PQL_MWSQUOTA	60 pages	4000 pagelets
PQL_DWSEXTENT	400 pages	12000 pagelets
PQL_MWSEXTENT	60 pages	4000 pagelets
PQL_DENQLM	30 locks	64 locks

Note

SYSGEN lists the DUMPSTYLE default value as 0, the same value as on a VAX system. A value of 0 specifies that the entire contents of physical memory will be written to the dump file. However, on OpenVMS AXP the AUTOGEN calculated value is 1. A value of 1 specifies that the contents of memory will be written to the dump file selectively to maximize the utility of the dump file while conserving disk space.

5.2.1 System Parameters and Memory-Intensive Applications

Please note the following about the system parameters on OpenVMS AXP:

- The WSINC (working set increment) system parameter sets the size in pagelets (pages, internally) to increase the working set size to compensate for a high page-fault rate. WSINC is dynamic, which means that you can alter its value without having to reboot the system. You might want to experiment with larger WSINC values. A large WSINC value might have a positive effect on the performance of memory-intensive applications.
- Also consider experimenting with the WSEXTENT process quota for processes that use memory-intensive applications. As noted in Chapter 2, the default value for the WSEXTENT process quota on VAX systems is increased for AXP systems. On VAX systems, the WSEXTENT default value is 512 512-byte pages. On AXP systems, the WSEXTENT default value is 16384 512-byte pagelets (1024 8192-byte pages). Even higher WSEXTENT process quota values might be appropriate on AXP computers with large amounts of memory, such as 256MB.

  You should avoid situations in which the WSEXTENT process quota or the WSINC system parameter limits the required (and available) working set memory.

- The PFRATH (page fault rate's high threshold) system parameter sets the upper threshold for the page fault rate for incrementing automatic working set size. On VAX systems, the PFRATH default value is 120 page faults every 10 seconds. On AXP systems, the PFRATH default value is reduced to 8 page faults every 10 seconds. The reduction was made on AXP systems because of the larger page size.
- AUTOGEN calculates values for PFCDEFAULT, WSINC and PFRATH at the end of the upgrade or installation. AUTOGEN attempts to set the values for those system parameters whether or not the INITIAL keyword was specified.
- Some application configurations (for example, a large number of memory-intensive processes) may benefit from a reduction in the AWSTIME and QUANTUM system parameters. AWSTIME specifies the minimum amount of processor time that must elapse for the system to collect a significant sample of a working set's page fault rate. The time is expressed in units of 10 milliseconds. The default value of 20, for example, is 200 milliseconds. QUANTUM sets the maximum amount of processor time a process can receive while other processes are waiting; the default value is 20 (200 milliseconds).

  The value for both parameters should be identical and can be as low as 4 (40 milliseconds).

5.3 Use of Page or Pagelet Values in Utilities and Commands

Section 1.1.1 describes the relative sizes of a VAX page (512 bytes), an AXP pagelet (also 512 bytes), and a CPU-specific AXP page (8192 bytes on an AXP computer with 8KB pages). Section 2.2.25 explains how these differences affect SYSUAF.DAT process quotas and their default values, and Section 5.1 and Section 5.2 explain the impact on system parameters.

In addition to process quotas and system parameters, the page and pagelet units affect other OpenVMS system management utilities and commands, as explained in the following list:

- SHOW MEMORY

  The Physical Memory Usage and Granularity Hint Regions statistics are shown in CPU-specific page units. Also, the Paging File Usage statistics are shown in blocks (rather than in 512-byte pages, as on VAX). For example:

```
$ SHOW MEMORY

              System Memory Resources on 24-AUG-1993 11:03:03.36

Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (96.00Mb)            12288        9714        2309         265

Granularity Hint Regions (pages):  Total        Free      In Use    Released
  Code region                        512           0         498          14
  Data region (User Read)            128           5          75          48
  Data region (Exec Read)            160           0         160           0

Slot Usage (slots):                Total        Free    Resident     Swapped
  Process Entry Slots                 80          62          18           0
  Balance Set Slots                   78          62          16           0

Dynamic Memory Usage (bytes):      Total        Free      In Use     Largest
  Nonpaged Dynamic Memory         925696       94720      830976        2624
  Paged Dynamic Memory            868352      519424      348928      518128
```

```
Paging File Usage (blocks):                         Free  Reservable       Total
  DISK$X5EE-D3A:[SYS0.SYSEXE]SWAPFILE.SYS          10112       10112       10112
  DISK$X5EE-D3A:[SYS0.SYSEXE]PAGEFILE.SYS         204544      164544      204544

Of the physical pages in use, 1261 pages are permanently allocated to VMS.

The VMS Cache is DISABLED on this node.
```

- SHOW SYSTEM

  On OpenVMS VAX systems, the SHOW SYSTEM output's rightmost column, Ph.Mem, shows the physical working set in 512-byte VAX page units. On OpenVMS AXP V1.5 systems, the SHOW SYSTEM/FULL command displays CPU-specific pages *and kilobytes* in the rightmost column, Pages. For example:

```
$ ! On an AXP node
$ SHOW SYSTEM/FULL/NODE=VAXCPU

OpenVMS V5.5-2  on node VAXCPU  22-AUG-1993 13:02:59.33   Uptime 12 19:46:39
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
2180501A DMILLER         HIB      8      310  0 00:00:03.91      1313   307
         [VMS,DMILLER]                                                  153Kb
21801548 _RTA1:          LEF      4       59  0 00:00:00.85       373   272
         [TEST_OF_LONG_USER_IDENTIFIERS_G,TEST_OF_LONG_USER_IDENTIFIER 136Kb
```

  Notes on the previous example:

  - One kilobyte (KB) equals 1024 bytes. Because the previous SHOW SYSTEM/FULL command displays pages from a VAX node's processes (where each of the 307 pages equals 512 bytes, or half of 1024) with /NODE, the utility halves the 307 pages for a resulting value of 153.5KB. This value is truncated to 153KB.
  - The second line for each process is displayed only when the /FULL qualifier is specified.
  - Long user identifiers are truncated to 61 characters, from a maximum of 65.

  The Pages column also shows CPU-specific pages and kilobytes when you use SHOW SYSTEM/FULL on an AXP node and you are displaying process information from the same or another AXP node in the VMScluster. In this case, each 8192-byte page equals 8KB. If the SHOW SYSTEM/FULL command displayed information about a process with 221 AXP pages, the value beneath it would be 1768KB (221*8).

  The SHOW SYSTEM command on OpenVMS AXP displays "OpenVMS" and the version number in the banner and never displays "VAX" or "AXP."

  In the following example, the SHOW SYSTEM/NODE command is executed on node AXPCPU and displays information about another AXP node in the VMScluster, AXPTOO:

```
$ SHOW SYSTEM/NODE=AXPTOO

OpenVMS V1.5 on node AXPTOO 22-AUG-1993 13:04:29.17  Uptime  ...
```

  In the following example, the SHOW SYSTEM/NODE command is executed on node AXPCPU and displays information about a VAX node (VAXCPU) running VMS Version 5.5–2 in the VMScluster:

```
$ ! On an AXP node
$ SHOW SYSTEM/NODE=VAXCPU

OpenVMS V5.5-2 on node VAXCPU 22-AUG-1993 13:06:51.23  Uptime  ...
```

In the following example, the SHOW SYSTEM/NODE command is executed on a VMS Version 5.5–2 node and displays information about an AXP node (AXPCPU) running OpenVMS Version 1.5 in a dual-architecture VMScluster. Because the VMS Version 5.5–2 SHOW SYSTEM/NODE command always displays "VAX/VMS" and the version number of the target system, you will notice the peculiar display "VAX/VMS V1.5":

```
$ ! On a VAX node, V5.5-2
$ SHOW SYSTEM/NODE=AXPCPU

VAX/VMS V1.5 on node AXPCPU 22-AUG-1993 13:09:22.45  Uptime  ...
```

- SHOW PROCESS/CONTINUOUS

  The Working set and Virtual pages columns show data in CPU-specific page units. The following edited output shows a snapshot of the display from a SHOW PROCESS/CONTINUOUS command:

```
$ SHOW PROCESS/CONTINUOUS

                         Process SMART                      09:52:11

    State               CUR                 Working set              108

    Cur/base priority   6/4                 Virtual pages            447
    .
    .
    .
    $65$DKB0:[SYS0.SYSCOMMON.][SYSEXE]SHOW.EXE
```

- MONITOR PROCESSES

  The PAGES column displays data in CPU-specific page units. For example:

```
$ MONITOR PROCESSES

Process Count: 26             VMS Monitor Utility      Uptime:  16 21:26:35
                                   PROCESSES
                                 on node SAMPLE
                              17-NOV-1993 09:58:48

        PID   STATE PRI   NAME         PAGES      DIOCNT FAULTS  CPU TIME

     2D000101 HIB   16 SWAPPER         0/0             0       0 00:00:00.5
     2D000105 HIB   10 CONFIGURE       0/22           10      36 00:28:30.1
     2D000106 HIB    7 ERRFMT          0/49         7907      35 00:00:21.8
     2D000107 HIB   16 CACHE_SERVER    0/31          468      22 00:00:00.2
     2D00010A HIB   10 AUDIT_SERVER    11/86         130     172 00:00:02.5
    .
    .
    .
```

- Ctrl/T key sequence

  The MEM field displays the current physical memory for the interactive process in CPU-specific page units. For example:

```
$ Ctrl T

SAMPLE::SMART 10:01:44   (DCL)   CPU=00:00:08.88 PF=5446 IO=4702 MEM=896
```

- SHOW PROCESS/ALL

  Under the Process Quotas category, the Paging file quota value is in pagelet units.

  Under the Accounting Information category, Peak working set size and Peak virtual size are in pagelet units.

Under the Process Dynamic Memory Area category, “Current Size (pagelets)” is in pagelet units.

For example:

```
$ SHOW PROCESS/ALL

17-NOV-1993 09:55:47.37   User: SMART            Process ID:   2D000215
                          Node: SAMPLE           Process name: "SMART"
   .
   .
   .
Process Quotas:
 Account name: VMS
 CPU limit:                      Infinite  Direct I/O limit:        100
 Buffered I/O byte count quota:     99808  Buffered I/O limit:      100
 Timer queue entry quota:              10  Open file quota:          99
 Paging file quota:                 98272  Subprocess quota:         10
 Default page fault cluster:           64  AST quota:                98
 Enqueue quota:                       600  Shared file limit:         0
 Max detached processes:                0  Max active jobs:           0

Accounting information:
 Buffered I/O count:      4059  Peak working set size:       3952
 Direct I/O count:         380  Peak virtual size:          16688
 Page faults:             5017  Mounted volumes:                0
 Images activated:          63
 Elapsed CPU time:          0 00:00:08.19
 Connect time:              5 18:35:37.35
   .
   .
   .
Process Dynamic Memory Area
    Current Size (bytes)         57344    Current Size (pagelets)         112
    Free Space (bytes)           40940    Space in Use (bytes)          16404
    Size of Largest Block        40812    Size of Smallest Block            8
    Number of Free Blocks            6    Free Blocks LEQU 64 Bytes         5
   .
   .
   .
```

- SET WORKING_SET/QUOTA

  The working set quota value that you can specify on the command line is in pagelet units. For example:

  ```
  $ SET WORKING_SET/QUOTA=6400

  %SET-I-NEWLIMS, new working set:   Limit = 150  Quota = 6400  Extent = 700
  ```

- SHOW WORKING_SET

  The displayed working set values for /Limit, /Quota, /Extent, Authorized Quota, and Authorized Extent are in pagelet units *and in CPU-specific page units:*

  ```
  $ SHOW WORKING_SET

    Working Set (pagelets)  /Limit=2000  /Quota=4000  /Extent=6000
    Adjustment enabled      Authorized Quota=4000  Authorized Extent=6000

    Working Set (8Kb pages) /Limit=125  /Quota=250  /Extent=375
                            Authorized Quota=250  Authorized Extent=375
  ```

  Page units are shown in addition to the pagelet units because SHOW PROCESS and MONITOR PROCESSES commands display CPU-specific pages.

- The following command qualifiers accept pagelet values:
  - RUN (process) /EXTENT /PAGE_FILE /WORKING_SET /MAXIMUM_WORKING_SET
  - SET QUEUE /WSDEFAULT /WSEXTENT /WSQUOTA
  - INITIALIZE/QUEUE /WSDEFAULT /WSEXTENT /WSQUOTA
  - START/QUEUE /WSDEFAULT /WSEXTENT /WSQUOTA
  - SET ENTRY /WSDEFAULT /WSEXTENT /WSQUOTA
  - SUBMIT /WSDEFAULT /WSEXTENT /WSQUOTA

When you or users on the AXP computer assign pagelets to the appropriate DCL command qualifiers, keep in mind the previously stated caveats—as noted in Section 2.2.25 and Section 5.1—about how pagelet values that are not multiples of 16 (on an AXP computer with 8KB pages) are rounded up to whole AXP pages.

5.4 Adaptive Pool Management

Adaptive pool management offers the following advantages:

- Simplified system management
- Improved performance
- Reduced overall pool memory requirements and less frequent denial of services because of exhaustion of resources

Note

Adaptive pool management is available on systems running OpenVMS AXP or OpenVMS VAX Version 6.0 and later releases. The features are not available on systems running VMS Version 5.5 and earlier releases.

Adaptive pool management provides dynamic lookaside lists plus reclamation policies for returning excess packets from the list to general nonpaged pool. Note that:

- Functional interfaces to existing routines remain unchanged.
- The basic nonpaged pool granularity is increased to 64 bytes. This quantity is justified by performance studies that show it to be the optimal granularity. This increase makes the effective natural alignment 64 bytes. The consumer of nonpaged pool can continue to assume 16-byte natural alignment.
- There are 80 lookaside lists spanning an allocation range of 1 to 5120 bytes in 64-byte increments. The lists are not prepopulated; that is, they start out empty. When an allocation for a given list's size fails because the list is empty, allocation from general pool occurs. When the packet is deallocated, it is added to the lookaside list for that packet's size. Thus, the lookaside lists self-populate over time to the level needed by the average work load on the system.
- The lookaside lists have a higher hit rate because of the increased number of sizes to which requests must be matched. The OpenVMS AXP method incurs 5 to 10 times fewer requests to general pool than the VAX VMS Version 5.*n* method.

- Adaptive pool management eliminates the four separate virtual regions of system space for nonpaged pool (three for lookaside lists, one for general pool). Instead, there is one large virtual region. The lookaside lists are populated from within this large region. A packet might be in one of the following states:
  - Allocated
  - Free in general pool
  - Free on some lookaside lists
- Overall memory consumption is expected to be approximately 5% less than with the VAX VMS Version 5.*n* method.

  "Gentle" reclamation keeps the lists from growing too big as the result of peaks in system usage. Every 30 seconds, each of the 80 lookaside lists is examined. For each one that has at least two entries, one entry is removed to a scratch list. After each scan, a maximum of 80 entries are in the scratch list, one from each list. The scratch list entries are then returned to general pool.

  "Aggressive" reclamation is triggered as a final effort to avoid pool extension from the variable allocation routine EXE$ALONPAGVAR. The lookaside list does not have to contain at least two entries to have one removed in the scan. Even if removal would leave the list empty, the entry is removed.

The adaptive pool management feature maintains usage statistics and extends detection of pool corruption. Two versions of the SYSTEM_PRIMITIVES executive image are provided that give you a boot-time choice of a minimal pool-code version or a pool-code version that features statistics and corruption detection:

- POOLCHECK zero (default value)

  SYSTEM_PRIMITIVES_MIN.EXE is loaded.
- POOLCHECK nonzero

  SYSTEM_PRIMITIVES.EXE, pool checking, and monitoring version are loaded.

With the pool monitoring version loaded, the pool management code maintains a ring buffer of the most recent 256 nonpaged pool allocation and deallocation requests. Two new SDA commands are added. The following command displays the history buffer:

```
SDA> SHOW POOL/RING_BUFFER
```

The following command displays the statistics for each lookaside list:

```
SDA> SHOW POOL/STATISTICS
```

With the addition of these two SDA commands, the MONITOR POOL command is not provided on OpenVMS AXP or on OpenVMS VAX Version 6.0.

5.5 Installing Main Images and Shareable Images In GHRs

On OpenVMS AXP, you can improve the performance of main images and shareable images that have been linked with /SHARE and a new LINK qualifier, /SECTION_BINDING=CODE, by installing them as resident with the Install utility (INSTALL). The code sections of an installed resident image reside in sections of memory consisting of multiple pages mapped by a single page table entry. These sections are known as granularity hint regions (GHRs). The AXP hardware can consider a set of pages as a single GHR because the GHR can be mapped by a single page table entry (PTE) in the translation buffer (TB). The result is a reduction in TB miss rates. Consequently, TBs on AXP systems are used more efficiently than if the loadable executive images or the shareable images were loaded in the traditional manner.

Also, the OpenVMS operating system executive images are, by default, loaded into GHRs. The result is that overall OpenVMS system performance is improved. This feature is controlled by a system parameter, as discussed in Section A.2.5.4.

These options are not available on OpenVMS VAX systems.

The GHR feature lets OpenVMS split the contents of images and sort the pieces so that they can be placed with other pieces that have the same page protection in the same area of memory. This method enables a single PTE to map the multiple pages.

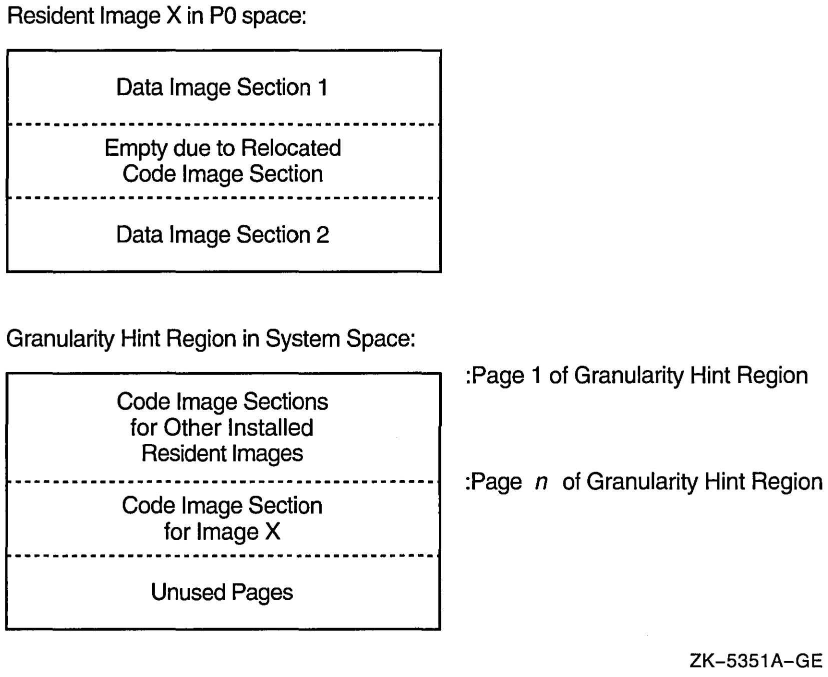
Application programmers are the likely users of the GHR feature for shareable images. As system manager, you might be asked to coordinate or assist GHR setup efforts by entering Install utility and SYSGEN utility commands.

The CODE keyword in the LINK/SECTION_BINDING=*option* command indicates that the linker should not optimize calls between code image sections by using a relative branch instruction. (See Section 5.5.2 for information about the DATA option to the /SECTION_BINDING qualifier.)

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

Unlike a loadable executive image, in which the entire image can be loaded into a GHR, only the code sections of an installed resident image are loaded into GHRs with granularity hints set. Data image sections must remain private to the process. Figure 5–1 shows what the virtual address space might look like for an installed resident image.

Figure 5–1 Virtual Address Space for Sample Installed Resident Image


Resident Image X in P0 space:
Data Image Section 1
Empty due to Relocated Code Image Section
Data Image Section 2
Granularity Hint Region in System Space:
Code Image Sections for Other Installed Resident Images
Code Image Section for Image X
Unused Pages
:Page 1 of Granularity Hint Region
:Page n of Granularity Hint Region
ZK-5351A-GE

5.5.1 Install Utility Support

Several OpenVMS AXP Install utility (INSTALL) commands have been enhanced to support the GHR feature. The ADD, CREATE, and REPLACE commands have a new qualifier, /RESIDENT. When this qualifier is specified, INSTALL loads all image sections that have the EXE and NOWRT attributes into one of the granularity hint regions (GHRs) in system space. If no image sections have these attributes, INSTALL issues the following warning message, where X is the image name:

```
%INSTALL-W-NORESSEC, no resident sections created for X
```

Note

Use of /RESIDENT is applicable only to loading main images or shareable images.

INSTALL can generate two error messages during attempts to install an image resident:

- If the image was not linked with the /SECTION_BINDING=CODE qualifier, the following error message is issued:

  ```
  %INSTALL-E-MISSLNKQUAL, image was not linked with /SECTION_BINDING=CODE
  ```

- If memory is not available in any of the GHRs, the following error message is issued, where X is the image name:

  ```
  %INSTALL-E-FAIL, failed to create entry for X
  -SYSTEM-F-INSFMEM, insufficient dynamic memory
  ```

The display produced by the INSTALL commands LIST and LIST/FULL shows those images that are installed resident. For the LIST/FULL command, the display shows how many resident code sections were found. For example:

```
INSTALL> LIST SYS$LIBRARY:FOO.EXE
   FOO;1    Open Hdr Shar        Lnkbl                 Resid

INSTALL> LIST/FULL
   FOO;1    Open Hdr Shar        Lnkbl                 Resid
        Entry access count       = 0
        Current / Maximum shared = 1 / 0
        Global section count     = 1
        Resident section count   = 0001
```

Note

The LIBOTS.EXE, LIBRTL.EXE, DPML$SHR.EXE, and DECC$SHR.EXE images are installed resident on OpenVMS AXP.

5.5.2 Image Activator Support

The image activator has been enhanced to support installed resident images. The fixup and relocation code must take this different layout of a process's address space into account.

One more optional feature is available for installed resident images. In Figure 5–1, notice that some amount of virtual address space is wasted as a result of the code image sections being located in the granularity hint region (GHR). You can eliminate this wasted space by compressing the data image sections, thereby making the data sections contiguous with each other.

To do this, an image must be linked with the DATA keyword specified with the /SECTION_BINDING qualifier, along with the CODE keyword. The DATA keyword indicates that the linker must make sure that no relative references exist between data image sections. See the *OpenVMS Linker Utility Manual* for details about this qualifier.

You can use the ANALYZE/IMAGE command on an OpenVMS AXP system to determine whether an image was linked with /SECTION_BINDING=(CODE,DATA). In the ANALYZE/IMAGE output, look for the EIHD$V_BIND_CODE and EIHD$V_BIND_DATA symbols; a value of 1 for each symbol indicates that /SECTION_BINDING=(CODE,DATA) was used.

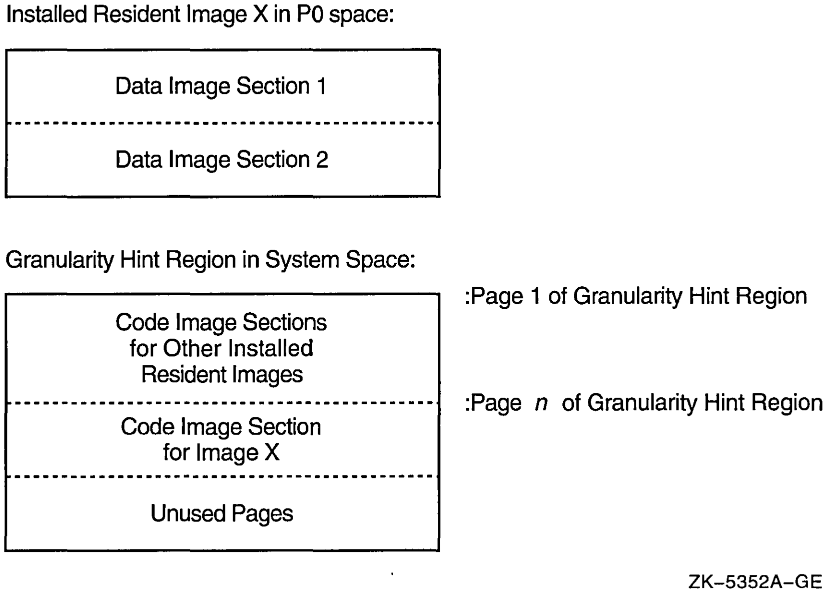
If the image is linked in this manner, the image activator compresses the data image sections in order not to waste virtual address space.

Note

While it does save virtual address space, the DATA option to the /SECTION_BINDING qualifier does not improve performance.

Figure 5–2 shows the virtual address space layout of image X from Figure 5–1 when it has been linked with the /SECTION_BINDING=(CODE,DATA) and /SHARE qualifiers.

Figure 5–2 Sample Installed Resident Image in a GHR


Installed Resident Image X in P0 space:
Data Image Section 1
Data Image Section 2
Granularity Hint Region in System Space:
Code Image Sections for Other Installed Resident Images
Code Image Section for Image X
Unused Pages
:Page 1 of Granularity Hint Region
:Page n of Granularity Hint Region
ZK−5352A−GE

5.5.3 System Parameter Support

Two new system parameters are associated with the GHR feature: ITB_ENTRIES and GH_RSRVPGCNT.

The ITB_ENTRIES parameter specifies the number of granularity hint regions (GHRs) usable by OpenVMS AXP. The default value for ITB_ENTRIES is 1. This GHR is used by the system loader to load executive images. It can also be used for installed resident images. If another GHR is needed or if GHR is disabled, then ITB_ENTRIES must be set to an appropriate value. For the initial AXP chip, a GHR is 4MB (512 pages), because each of the four ITB entries that support granularity hints only support a hint of 512 pages.

The GH_RSRVPGCNT parameter specifies the number of unused pages within a GHR to be retained after startup. The default value for GH_RSRVPGCNT is 0. At the end of startup, the LDR$WRAPUP.EXE image executes and releases all unused portions of the GHR except for the amount specified by GH_RSRVPGCNT, assuming that the SGN$V_RELEASE_PFNS flag is set in the system parameter LOAD_SYS_IMAGES. This flag is set by default. Setting GH_RSRVPGCNT to a nonzero value lets images be installed resident at run time.

If there are no GH_RSRVPGCNT pages in the GHR when LDR$WRAPUP runs, no attempt is made to allocate more memory. Whatever free space is left in the GHR will be available for use by INSTALL.

5.5.4 AUTOGEN Support

AUTOGEN has been modified to update the GH_RSRVPGCNT and ITB_ENTRIES system parameters if feedback is specified or if these parameters are explicitly specified in the MODPARAMS.DAT file.

When feedback is requested, the GH_RSRVPGCNT parameter is modified based on how many pages were allocated from the granularity hint code region after system startup has completed. AUTOGEN will set GH_RSRVPGCNT to the smaller value of this number of pages (pages allocated after system startup) plus an additional 10% in order to leave room for either expansion or its current value.

After AUTOGEN has calculated a new value for GH_RSRVPGCNT, it will then modify ITB_ENTRIES accordingly.

AUTOGEN updates the GH_RSRVPGCNT and ITB_ENTRIES parameters unless any of the following conditions exist:

- Feedback is not specified and the GH_RSRVPGCNT and ITB_ENTRIES parameters are not explicitly set in MODPARAMS.DAT.
- Feedback is specified, but the SGN$V_RELEASE_PFNS flag is not set in the LOAD_SYS_IMAGES system parameter, which indicates that no page-frame numbers (PFNs) are to be released from the granularity hint regions. Therefore, there is no need to modify these system parameters.
- Feedback is specified, but the number of pages currently allocated in the granularity hint code region is less than the number of pages allocated at the end of system startup.

5.5.5 SHOW MEMORY Support

The DCL command SHOW MEMORY has been enhanced to support the granularity hint region (GHR) feature. The new /GH_REGIONS qualifier displays information about the GHRs that have been established. For each of these regions, information is displayed about the size of the region, the amount of free memory, the amount of memory in use, and the amount of memory released from the region.

In addition, the GHR information is displayed as part of the SHOW MEMORY, SHOW MEMORY/ALL, and SHOW MEMORY/FULL commands.

5.5.6 Loader Changes for Executive Images in GHRs

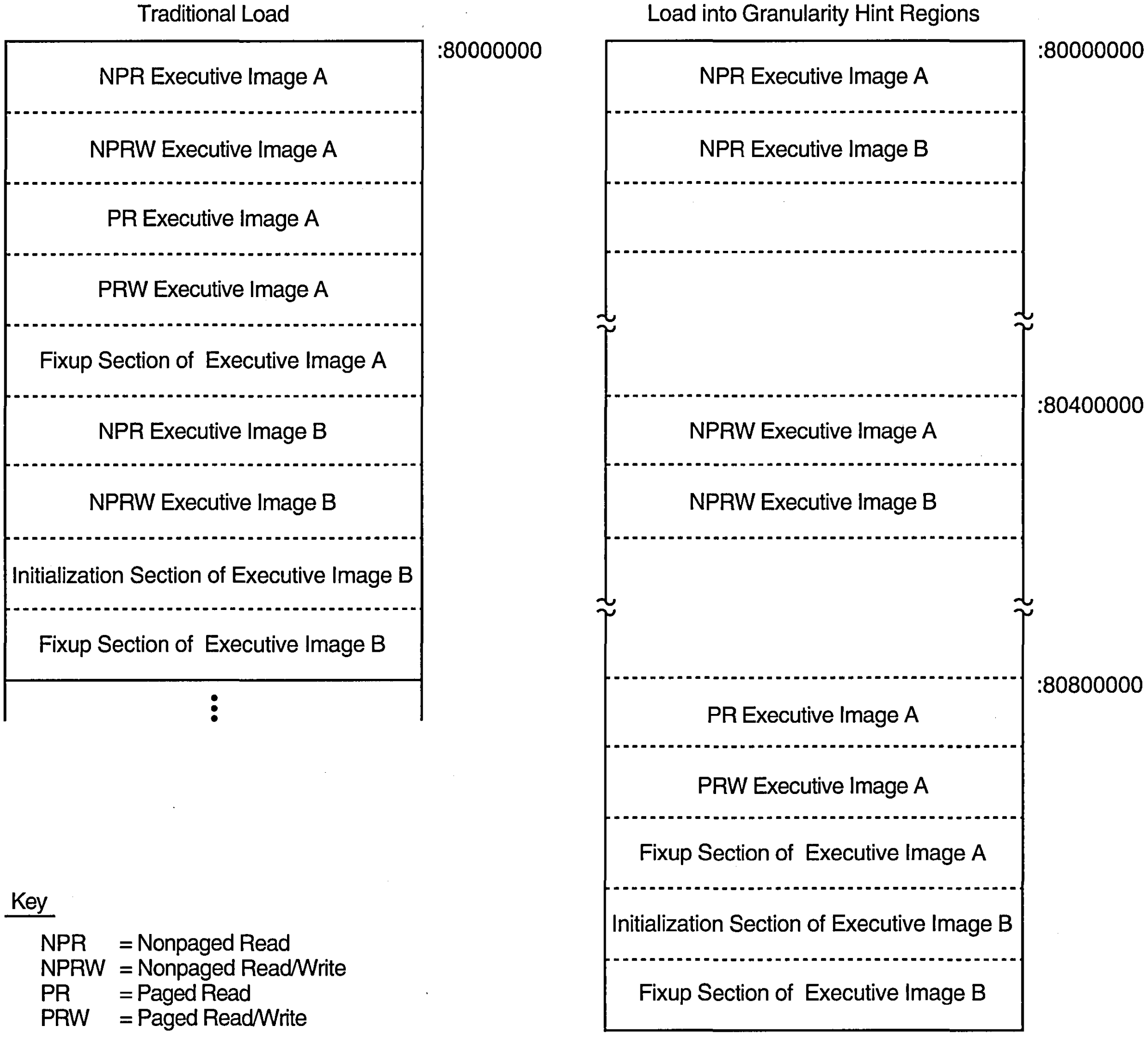
In traditional executive image loading, code and data are sparsely laid out in system address space. The loader allocates the virtual address space for executive images so that the image sections are loaded on the same boundaries as the linker created them.

The loader allocates a granularity hint region (GHR) for nonpaged code and another for nonpaged data. Pages within a GHR must have the same protection; hence, code and data cannot share a GHR. The end result is a single TB entry to map the executive nonpaged code and another to map the nonpaged data.

The loader then loads like nonpaged sections from each loadable executive image into the same region of virtual memory. The loader ignores the page size with which the image was linked. Paged, fixup, and initialization sections are loaded in the same manner as the traditional loader. If the S0_PAGING parameter is set to turn off paging of the executive image, all code and data, both paged and nonpaged, are treated as nonpaged and are loaded into the GHR.

Figure 5–3 illustrates a traditional load and a load into a GHR.

Figure 5–3 Traditional Loads and Loads into GHRs



ZK–5353A–GE

5.6 Virtual I/O Cache

The virtual I/O cache is a file-oriented disk cache that reduces I/O bottlenecks and improves performance. Cache operation is transparent to application software and requires little system management. This new functionality provides a write-through cache that maintains the integrity of disk writes while significantly improving read performance.

The virtual I/O cache works only on single-node systems. The function is disabled in a VMScluster environment.

By default, virtual I/O caching is enabled. To disable caching, set the system parameter VCC_FLAGS to 0; to enable it again, set the parameter to 1. By default, memory is allocated for caching 6400 disk blocks. This requires 3.2MB of memory. Use the VCC_MAXSIZE system parameter to control memory allocation.

Use the DCL commands SHOW MEMORY/CACHE and SHOW MEMORY /CACHE/FULL to observe cache statistics.

6 Network Management Tasks

Use DECnet for OpenVMS AXP Version 1.5 to establish networking connections with other nodes. DECnet for OpenVMS, previously known as DECnet–VAX, implements Phase IV of DNA. The DECnet for OpenVMS AXP features are similar to those of the DECnet–VAX software that is part of VMS Version 5.4–3, with a few exceptions. This chapter:

- Identifies which OpenVMS network management tasks remain the same on AXP and VAX systems.
- Explains how some network management tasks differ on OpenVMS AXP.

6.1 Network Management Tasks That Are the Same

Table 6–1 lists the OpenVMS network management tasks that are identical or similar on AXP and VAX computers.

Table 6–1 Identical or Similar OpenVMS Network Management Tasks

Feature or Task	Comment
Product Authorization Key (PAK) names	The PAK name for the end node license (DVNETEND) is the same on AXP and VAX systems. However, the PAK name enabling cluster alias routing support on OpenVMS AXP (DVNETEXT) is different from the PAK name enabling cluster alias routing support on OpenVMS VAX (DVNETRTG). See Section 6.2.1 for related information.
Cluster alias	Similar; however, level 1 routing is supported only on routers for a cluster alias. See Section 6.2.1 for more information.
NETCONFIG.COM procedure	The same, with one exception. See Section 6.2.1 for more information.
NETCONFIG_UPDATE.COM procedure	Identical.
Configuring DECnet databases and starting OpenVMS AXP computer's access to the network	The process of configuring your node and starting DECnet network connections for your computer is essentially the same on OpenVMS AXP (with the limitations in Section 6.2 in mind). The functions of the SYS$MANAGER:STARTNET.COM procedure are similar.

(continued on next page)

Table 6–1 (Cont.) Identical or Similar OpenVMS Network Management Tasks

Feature or Task	Comment
File access listener (FAL)	FAL is fully compatible with the OpenVMS VAX version. For example, bidirectional file transfer via the COPY command, which uses the FAL object, is identical.
Maximum network size	The same Phase IV limitations for VAX nodes running DECnet for OpenVMS (1023 nodes per area and 63 areas in the entire network).
Node name rules	The same rules and 6-character maximum length as with OpenVMS VAX nodes running DECnet for OpenVMS.
DECnet objects	Identical.
Task-to-task communication	Identical.
Network management via Network Control Program (NCP) utility and the network management listener (NML) object	In many cases, the NCP commands and parameters are identical. However, a number of NCP command parameters that are available for SET and subsequent SHOW operations have no effect on OpenVMS AXP. This characteristic is due to the lack of support for DDCMP, full host-based routing, the Distributed Name Service (DNS), and VAX P.S.I. See Section 6.2.5 for details.
Event logging	Identical.
DECnet Test Sender/DECnet Test Receiver utility (DTS/DTR)	Identical.
Downline load and upline dump operations	Identical.
Loopback mirror testing	Identical.
Ethernet monitor (NICONFIG)	Identical.
Supported lines	Ethernet and FDDI.
Routing	Level 1 routing is supported only on nodes acting as routers for a cluster alias. Level 2 routing and routing between multiple circuits is not supported. See Section 6.2.1 and Section 6.2.5 for details.
SET HOST capabilities	Identical.

6.2 Network Management Tasks That Are Different

This section describes the VMS network management tasks that are different on AXP systems. The differences are:

- Level 2 routing (between DECnet areas) is not supported. Level 1 routing is supported only on nodes acting as routers for a cluster alias. Routing between multiple circuits is not supported. Section 6.2.1.
- Some line types are not supported. See Section 6.2.2.
- The Distributed Name Service (DNS) node name interface that is used on OpenVMS VAX is not supported on OpenVMS AXP. See Section 6.2.3.
- VAX P.S.I. is not supported. See Section 6.2.4.

- A number of NCP command parameters are affected by unsupported features. See Section 6.2.5 for details.
- DECnet/OSI for OpenVMS is not supported on AXP systems at this time. See Section 6.2.6.

6.2.1 Level 1 Routing Supported for Cluster Alias Only

Level 1 DECnet routing is available, but is supported only on DECnet for OpenVMS AXP nodes acting as routers for a cluster alias. Routing between multiple circuits is not supported. Level 2 routing is not supported on DECnet for OpenVMS AXP nodes.

Note that the Product Authorization Key (PAK) name enabling DECnet for OpenVMS AXP cluster alias routing support (DVNETEXT) is different from the PAK name enabling cluster alias routing support on OpenVMS VAX (DVNETRTG). The functions supported with the DVNETEXT license differ from the DVNETRTG license. DVNETEXT is supported only to enable level 1 routing on AXP nodes acting as routers for a cluster alias.

Enabling a cluster alias requires differing steps, depending on whether you want the alias enabled for incoming, outgoing, or both types of connections. The different cases are documented in the *DECnet for OpenVMS Networking Manual* and *DECnet for OpenVMS Network Management Utilities.*

With the cluster alias feature, the difference between AXP and VAX systems is that you will have to manually enable level 1 routing on one of the AXP nodes[1] because the NETCONFIG.COM command procedure does not ask the routing question. (On VAX systems, NETCONFIG.COM prompts the user with the query, "Do you want to operate as a router?")

Enter DEFINE commands in NCP that are similar to the following example. (If DECnet is already running, shut down DECnet, enter the DEFINE commands, then restart DECnet.) The following example enables level 1 routing on one or more nodes, identifies the node name or node address of the alias node, and allows this node to accept (ENABLED) or not accept (DISABLED) incoming connections addressed to the cluster alias:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE EXECUTOR TYPE ROUTING IV
NCP>DEFINE EXECUTOR ALIAS NODE node-id node-address
NCP>DEFINE EXECUTOR ALIAS INCOMING {ENABLED DISABLED}
```

Specifying ENABLED or DISABLED with the NCP command DEFINE EXECUTOR ALIAS INCOMING is the network manager's choice, depending on whether you want this node to accept incoming connections directed to the cluster alias. If set to DISABLED, another node in the VMScluster having this parameter set to ENABLED will handle the connection.

Note that the CLUSTER_CONFIG.COM procedure uses NCP commands to configure the cluster alias when the following two conditions exist:

- You add a new VMScluster member node.
- The system from which you are running CLUSTER_CONFIG.COM has a cluster alias defined.

See the *DECnet for OpenVMS Networking Manual* and *DECnet for OpenVMS Network Management Utilities* for details.

[1] In a dual-architecture VMScluster, you do not have to enable level 1 routing on an AXP node if one of the VAX VMS Version 5.5–2 nodes is a routing node.

See Section 6.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS AXP systems because level 2 routing is not supported and level 1 routing is reserved for the cluster alias.

6.2.2 CI and DDCMP Lines Not Supported

OpenVMS AXP nodes can connect to a local area network (LAN) only via Ethernet lines or FDDI lines.

DECnet communication over CI lines is not supported. There also is no support for DDCMP lines.

Because DDCMP lines are not supported, the DCL command SET TERMINAL /PROTOCOL=DDCMP/SWITCH=DECNET also is not supported on OpenVMS AXP systems.

See Section 6.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS AXP systems because DDCMP is not supported.

6.2.3 DNS Node Name Interface Not Supported

The Distributed Name Service (DNS) node name interface that is used on OpenVMS VAX is not supported on OpenVMS AXP. Consequently, DNS object names cannot be specified by users and applications on OpenVMS AXP nodes. See Section 6.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS AXP systems because a DNS is not supported.

6.2.4 VAX P.S.I. Not Supported

VAX P.S.I., a software product that enables connections to X.25 and X.29 networks, is not supported. See Section 6.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS AXP systems because VAX P.S.I. is not supported.

Although VAX P.S.I. is not supported, an OpenVMS AXP node can connect to X.25 and X.29 networks via an X.25 or X.29 router on the same local area network.

6.2.5 NCP Command Parameters Affected by Unsupported Features

On nodes running DECnet for OpenVMS AXP, you can set unsupported NCP command parameters and then display those settings with the SHOW command. However, such parameters have no effect on OpenVMS AXP systems and are related to the following unsupported features or products:

- DDCMP
- VAX P.S.I. (Note that an OpenVMS AXP node can connect to X.25 and X.29 networks via an X.25 or X.29 router on the same local area network.)
- Level 2 routing (Level 1 routing is supported only on nodes acting as routers for a cluster alias; routing between multiple circuits is not supported.)
- Distributed Name Service (DNS)

Note

The characteristic on OpenVMS AXP of being able to set and show NCP command parameters that are related to unsupported features is similar to the same characteristic on OpenVMS VAX. For example, on a VAX system you could set and show NCP command parameters related to X.25 networks even if you had not installed VAX P.S.I.

Table 6–2 lists the affected NCP command parameters related to the unsupported DDCMP circuits, VAX P.S.I. software, full host-based routing, and the DNS node name interface. Refer to the *DECnet for OpenVMS Network Management Utilities* manual for details about how these parameters are used on OpenVMS VAX computers.

Table 6–2 NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
CLEAR/PURGE CIRCUIT ACTIVE BASE	DDCMP
CLEAR/PURGE CIRCUIT ACTIVE INCREMENT	DDCMP
CLEAR/PURGE CIRCUIT BABBLE TIMER	DDCMP
CLEAR/PURGE CIRCUIT DEAD THRESHOLD	DDCMP
CLEAR/PURGE CIRCUIT DYING BASE	DDCMP
CLEAR/PURGE CIRCUIT DYING INCREMENT	DDCMP
CLEAR/PURGE CIRCUIT DYING THRESHOLD	DDCMP
CLEAR/PURGE CIRCUIT INACTIVE BASE	DDCMP
CLEAR/PURGE CIRCUIT INACTIVE INCREMENT	DDCMP
CLEAR/PURGE CIRCUIT INACTIVE THRESHOLD	DDCMP
CLEAR/PURGE CIRCUIT MAXIMUM BUFFERS	DDCMP
CLEAR/PURGE CIRCUIT MAXIMUM RECALLS	DDCMP
CLEAR/PURGE CIRCUIT MAXIMUM TRANSMITS	DDCMP
CLEAR/PURGE CIRCUIT NETWORK	VAX P.S.I.
CLEAR/PURGE CIRCUIT NUMBER	VAX P.S.I.
CLEAR/PURGE CIRCUIT RECALL TIMER	VAX P.S.I.
CLEAR/PURGE CIRCUIT TRANSMIT TIMER	DDCMP
CLEAR/PURGE EXECUTOR AREA MAXIMUM COST	Host-based routing[1]
CLEAR/PURGE EXECUTOR AREA MAXIMUM HOPS	Host-based routing
CLEAR/PURGE EXECUTOR DNS INTERFACE	DNS node name interface
CLEAR/PURGE EXECUTOR DNS NAMESPACE	DNS node name interface
CLEAR/PURGE EXECUTOR IDP	DNS node name interface
CLEAR/PURGE EXECUTOR MAXIMUM AREA	Host-based routing
CLEAR/PURGE EXECUTOR MAXIMUM PATH SPLITS	Host-based routing

[1] Level 2 routing is not supported. Level 1 routing is supported only on nodes acting as routers for a cluster alias; routing between multiple circuits is not supported.

(continued on next page)

Table 6–2 (Cont.) NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
CLEAR/PURGE EXECUTOR PATH SPLIT POLICY	Host-based routing
CLEAR/PURGE EXECUTOR ROUTING TIMER	Host-based routing
CLEAR/PURGE EXECUTOR SUBADDRESSES	VAX P.S.I.
CLEAR/PURGE LINE DEAD TIMER	DDCMP
CLEAR/PURGE LINE DELAY TIMER	DDCMP
CLEAR/PURGE LINE HANGUP	DDCMP
CLEAR/PURGE LINE HOLDBACK TIMER	VAX P.S.I.
CLEAR/PURGE LINE LINE SPEED	DDCMP
CLEAR/PURGE LINE MAXIMUM RETRANSMITS	VAX P.S.I.
CLEAR/PURGE LINE SCHEDULING TIMER	DDCMP
CLEAR/PURGE LINE STREAM TIMER	DDCMP
CLEAR/PURGE LINE SWITCH	DDCMP
CLEAR/PURGE LINE TRANSMIT PIPELINE	DDCMP
CLEAR/PURGE MODULE X25-ACCESS (all parameters)	VAX P.S.I.
CLEAR/PURGE MODULE X25-PROTOCOL (all parameters)	VAX P.S.I.
CLEAR/PURGE MODULE X25-SERVER/X29-SERVER (all parameters)	VAX P.S.I.
CLEAR/PURGE NODE INBOUND	DDCMP
LOOP LINE (all parameters)	VAX P.S.I.
SET/DEFINE CIRCUIT ACTIVE BASE	DDCMP
SET/DEFINE CIRCUIT ACTIVE INCREMENT	DDCMP
SET/DEFINE CIRCUIT BABBLE TIMER	DDCMP
SET/DEFINE CIRCUIT CHANNEL	VAX P.S.I.
SET/DEFINE CIRCUIT DEAD THRESHOLD	DDCMP
SET/DEFINE CIRCUIT DTE	VAX P.S.I.
SET/DEFINE CIRCUIT DYING BASE	DDCMP
SET/DEFINE CIRCUIT DYING INCREMENT	DDCMP
SET/DEFINE CIRCUIT DYING THRESHOLD	DDCMP
SET/DEFINE CIRCUIT INACTIVE BASE	DDCMP
SET/DEFINE CIRCUIT INACTIVE INCREMENT	DDCMP
SET/DEFINE CIRCUIT INACTIVE THRESHOLD	DDCMP
SET/DEFINE CIRCUIT MAXIMUM BUFFERS	DDCMP
SET/DEFINE CIRCUIT MAXIMUM DATA	VAX P.S.I.
SET/DEFINE CIRCUIT MAXIMUM RECALLS	VAX P.S.I.
SET/DEFINE CIRCUIT MAXIMUM TRANSMITS	DDCMP
SET/DEFINE CIRCUIT MAXIMUM WINDOW	VAX P.S.I.

(continued on next page)

Table 6–2 (Cont.) NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
SET/DEFINE CIRCUIT NETWORK	VAX P.S.I.
SET/DEFINE CIRCUIT NUMBER	VAX P.S.I.
SET/DEFINE CIRCUIT OWNER EXECUTOR	VAX P.S.I.
SET/DEFINE CIRCUIT POLLING STATE	DDCMP
SET/DEFINE CIRCUIT RECALL TIMER	VAX P.S.I.
SET/DEFINE CIRCUIT TRANSMIT TIMER	DDCMP
SET/DEFINE CIRCUIT TRIBUTARY	DDCMP
SET/DEFINE CIRCUIT TYPE X25	VAX P.S.I.
SET/DEFINE CIRCUIT USAGE	VAX P.S.I.
SET/DEFINE CIRCUIT VERIFICATION	DDCMP
SET/DEFINE EXECUTOR AREA MAXIMUM COST	Host-based routing
SET/DEFINE EXECUTOR AREA MAXIMUM HOPS	Host-based routing
SET/DEFINE EXECUTOR DNS INTERFACE	DNS node name interface
SET/DEFINE EXECUTOR DNS NAMESPACE	DNS node name interface
SET/DEFINE EXECUTOR IDP	DNS node name interface
SET/DEFINE EXECUTOR MAXIMUM AREA	Host-based routing
SET/DEFINE EXECUTOR MAXIMUM PATH SPLITS	Host-based routing
SET/DEFINE EXECUTOR PATH SPLIT POLICY	Host-based routing
SET/DEFINE EXECUTOR ROUTING TIMER	Host-based routing
SET/DEFINE EXECUTOR SUBADDRESSES	VAX P.S.I.
SET/DEFINE EXECUTOR TYPE	The AREA node-type parameter is not supported because of the lack of level 2 host-based routing. The NONROUTING IV node-type parameter is supported. The ROUTING IV node-type parameter is supported for the node's cluster alias router only.
SET/DEFINE LINE CLOCK	DDCMP
SET/DEFINE LINE DEAD TIMER	DDCMP
SET/DEFINE LINE DELAY TIMER	DDCMP
SET/DEFINE LINE DUPLEX	DDCMP
SET/DEFINE LINE HANGUP	DDCMP
SET/DEFINE LINE HOLDBACK TIMER	VAX P.S.I.
SET/DEFINE LINE INTERFACE	VAX P.S.I.
SET/DEFINE LINE SPEED	DDCMP
SET/DEFINE LINE MAXIMUM BLOCK	VAX P.S.I.
SET/DEFINE LINE MAXIMUM RETRANSMITS	VAX P.S.I.

(continued on next page)

Table 6–2 (Cont.) NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
SET/DEFINE LINE MAXIMUM WINDOW	VAX P.S.I.
SET/DEFINE LINE MICROCODE DUMP	VAX P.S.I.
SET/DEFINE LINE NETWORK	VAX P.S.I.
SET/DEFINE LINE RETRANSMIT TIMER	DDCMP and VAX P.S.I.
SET/DEFINE LINE SCHEDULING TIMER	DDCMP
SET/DEFINE LINE STREAM TIMER	DDCMP
SET/DEFINE LINE SWITCH	DDCMP
SET/DEFINE LINE TRANSMIT PIPELINE	DDCMP
SET/DEFINE MODULE X25-ACCESS (all parameters)	VAX P.S.I.
SET/DEFINE MODULE X25-PROTOCOL (all parameters)	VAX P.S.I.
SET/DEFINE MODULE X25-SERVER/X29-SERVER (all parameters)	VAX P.S.I.
SET/DEFINE NODE INBOUND	DDCMP
SHOW/LIST CIRCUIT display: Polling substate value	DDCMP
SHOW/LIST MODULE X25-ACCESS (all parameters)	VAX P.S.I.
SHOW/LIST MODULE X25-PROTOCOL (all parameters)	VAX P.S.I.
SHOW/LIST MODULE X25-SERVER/X29-SERVER (all parameters)	VAX P.S.I.
ZERO MODULE X25-PROTOCOL (all parameters)	VAX P.S.I.
ZERO MODULE X25-SERVER/X29-SERVER (all parameters)	VAX P.S.I.

6.2.6 DECnet/OSI for OpenVMS

DECnet/OSI for OpenVMS is an OSI-compliant product. It is not available for AXP nodes at this time. DECnet/OSI for OpenVMS conforms to networking standards defined by the International Organization for Standardization (ISO). DECnet/OSI for OpenVMS features include:

- Much larger networks than the 64,449-node limit in Phase IV networks.
- A longer, more descriptive format for node names.
- Use of a local name database or a distributed namespace database.

A

Overview of OpenVMS AXP Performance Characteristics

Alpha AXP is a reduced instruction set computer (RISC) architecture. Computers using RISC architectures have substantially different characteristics from complex instruction set computer (CISC) architectures, such as VAX. The instruction set of a RISC computer is much simpler than the instruction set of a CISC system. In many cases, more RISC instructions are required to perform the equivalent operation of one CISC instruction. However, the RISC instruction set allows optimal customized compiler-generated code because it is often possible to load registers with desired values and perform most operations in the registers before accessing main memory. In these instances, the RISC machines, when coupled with sophisticated compilers, can provide significant performance improvements over CISC machines. High-level language compilers provide greater possibility for optimized code generation than low-level languages such as VAX MACRO or MACRO-64 for OpenVMS AXP. Therefore, Digital recommends that you use high-level languages when programming on OpenVMS AXP.

A.1 AXP Programming Considerations

Because the Alpha AXP architecture is significantly different from the VAX architecture, the code generation necessary for good performance is significantly different on OpenVMS AXP than on OpenVMS VAX. You can run translated VAX images directly on OpenVMS AXP with performance comparable to that of an equivalent technology VAX. In order to take advantage of the power of the AXP RISC system, you should recompile and relink the application as a native OpenVMS AXP image. When an application written in a high-level language is compiled and relinked for OpenVMS AXP, the programmer must be aware of certain considerations described in the following subsections. For specific information about porting applications to OpenVMS AXP, see the following manuals:

- *Migrating to an OpenVMS AXP System: Porting VAX MACRO Code*
- *Migrating to an OpenVMS AXP System: Planning for Migration*
- *Migrating to an OpenVMS AXP System: Recompiling and Relinking Applications*
- *DECmigrate for OpenVMS AXP Version 1.0 Translating Images*

A.1.1 Choosing a Programming Language

If possible, avoid programming in native MACRO-64 for OpenVMS AXP. The high-level language compilers generate highly optimized code that provides good performance on OpenVMS AXP.

If possible, avoid programming in VAX MACRO. VAX MACRO can be compiled on OpenVMS AXP. The compiler generates code that is optimized for OpenVMS AXP, but many features of VAX MACRO that provide the programmer with a high level of control make it more difficult to generate optimal code for OpenVMS AXP in the MACRO-32 environment. In general, the high-level languages provide more optimal code generation than the low-level languages.

A.1.2 Data Alignment

When optimal performance is necessary, align data structures and fields of data structures on natural longword or quadword boundaries—longword structures on longword addresses, quadword structures on quadword addresses, and so on. On OpenVMS AXP, the performance dividend from appropriate longword or quadword alignment of data cells is significant. Operations on nonnaturally aligned data on OpenVMS AXP systems can be up to 10 times slower than operations on aligned data.

All memory accesses on OpenVMS AXP must be aligned on longword or quadword boundaries. The smallest addressable item that can be read from memory is a longword. This characteristic has the following implications:

- If the address being retrieved or written is known by the compiler to be unaligned, extra instructions must be generated to perform the access.
- If the address is assumed to be aligned but in fact is not, an exception is generated and the access is fixed up.
- If a datum smaller than a longword is accessed, the compiler generates extra instructions to perform the byte manipulation on the surrounding aligned longword or quadword.

All of these situations decrease performance. Of these, the unaligned exception is by far the most serious and should be avoided.

The following are suggestions for improving performance:

- Align data structures naturally, that is, quadword fields on quadword boundaries, longword fields on longword boundaries, and word fields on word boundaries. Rearrange the structure or add filler fields if necessary to accomplish this.
- Force data structures to begin on quadword boundaries (longword boundaries are acceptable if there are no quadword elements in the data structure).
- Promote words and bytes to longwords. This is important for both elements of structures and single variables, and when a data cell is referenced frequently.

A.1.3 Simple and Complex Instructions

Privileged architecture library instructions (PALcode) provide ways to emulate certain VAX operations on AXP machines. A number of these operations are simple ones that involve synchronization. The synchronization may not be necessary, depending on what you are trying to do, and the relative performance impact might be high.

For example, consider the BBSSI instruction (VAX MACRO, BLISS built-in, or LIB$BBSSI routine). If you are using the instruction simply to set a bit and synchronization is not needed, using a different method such as a BISL instruction would be better.

Another area to consider is floating point usage. Occasionally, programs use floating point operations inconsistently because a particular expression is easier to write that way, or because the expression is the default. OpenVMS AXP has a separate set of floating point registers and, therefore, a separate floating point context. If you are using floating point instructions, then your application must save and restore the context; this process uses both space and CPU time. Unless you are already using floating point operations elsewhere in the program, it is better not to use floating point operations where integer operations could be used equivalently.

A.1.4 Code Path Length

One consistently questionable area in any migration to RISC is code path length. Because of the smaller RISC instruction set, more instructions are usually required to do complex operations.

Do not attach too much significance to this issue. Code path length is generally a poor indicator of performance. Register conflicts and memory delays can cause an apparently short code path to run slowly because of stalls. PALcode calls appearing within the path result in a performance penalty. On the other hand, a long code path that is well suited to dual-issue instruction processing and with few memory references can run very quickly.

A.1.5 Multiple Compilation Units and Code Optimization

When compiling multiple source modules, the compiler can provide substantially greater performance optimization when multiple modules are compiled into a single object module. You can do this by specifying to the compiler a list of source modules separated by plus signs (+) rather than by compiling each module individually and then joining the modules at link time. The basic rule is that the more the compiler knows about the source code, the more optimal the code it generates. Therefore, increasing the compiler's scope improves its ability to generate optimal code.

A.1.6 AXP Memory Size

Bear in mind that AXP machines generally have large amounts of memory, so saving memory space is not usually much of an issue. The performance gained by increasing the size of a variable to a longword or quadword is usually worth the extra space it occupies. Similarly, the performance gain from filling a structure in order to align elements on natural boundaries is also worth the extra space.

A.1.7 Summary of Differences Between Alpha AXP and VAX Architectures

The following list summarizes key differences between AXP and VAX systems:

- AXP RISC instructions are faster and simpler than VAX CISC instructions.
- Multiple AXP instructions can enter the instruction pipeline; VAX systems can issue only one at a time.
- On VAX systems, a single instruction can perform operations directly on memory; all AXP operations are performed on registers.
- On VAX systems, condition codes are set on each instruction; on AXP systems, explicit tests are required to determine a given condition.

- In most cases, multiple AXP RISC instructions are required to perform the equivalent function of a VAX CISC instruction.
- AXP RISC instructions take fewer cycles than VAX CISC instructions.
- In many cases, register usage can be optimized to a greater extent on RISC machines than on CISC machines because RISC instructions are lower-level instructions.
- On VAX systems, the smallest addressable unit is a byte; on AXP, the smallest addressable unit is a longword (4 bytes).
- On VAX systems, there are 16 longword registers; on AXP, there are 32 quadword (8 bytes) integer and 32 quadword floating point registers.
- AXP is based on a 64-bit architecture; VAX is based on a 32-bit architecture.
- On VAX systems, the CPU generally modifies the internal registers of I/O devices directly, making the CPU and I/O devices relatively tightly coupled.
- On many AXP systems, mailboxes are used to communicate with I/O devices, providing a looser coupling between the CPU and I/O devices, thereby allowing each subsystem to run with less dependence on the other and thus at greater speeds. (This statement does not apply to AXP workstations.)
- On VAX systems, every page requires a translation buffer (TB) entry to reference it; on AXP systems, granularity hints allow multiple physically contiguous pages to be referenced by a single TB entry.

A.2 AXP Performance Features

OpenVMS AXP has several features that provide improved performance: granularity hints, the zero page list, and image slicing of the OpenVMS AXP executive and user libraries.

A.2.1 Granularity Hints and the Translation Buffer

The translation buffer (TB) contains a group of entries that are used by the system to translate virtual addresses to physical addresses. The Alpha AXP architecture introduces a capability called a **granularity hint**, a value in the page table entry (PTE) that denotes how many pages to map beyond the page referenced in the PTE. Thus, a granularity hint allows a single TB entry to map multiple pages of like characteristics with a single TB entry. These pages are called a granularity hint region (GHR). Granularity hints allow more effective use of TB entries than can be accomplished on VAX systems because VAX requires a TB entry for each page. In general, TB entries derived from large granularity values are rarely removed from the TB, so a TB miss rarely occurs on a system page that is part of a granularity hint region.

Finally, TB entries using granularity hints map multiple pages, thereby leaving more TB entries for users. In combination with this, the AXP TB has an address space number (ASN) that allows TB entries for multiple processes to coexist in the TB at any point in time. ASNs allow TB entries for a process that is temporarily inactive to remain in the TB until the process becomes active again. A granularity hint region, therefore, increases the probability that a TB entry will remain in the TB for a longer period of time because the region decreases the demand for TB entries. The relative cost of a TB miss is substantially higher in OpenVMS AXP than on OpenVMS VAX, so minimizing the number of TB misses on OpenVMS AXP is important.

A.2.2 Zero Page List

OpenVMS AXP introduces a page list called the **zero page list**. Pages are substantially larger on AXP than on VAX. To zero a demand-zero page requires CPU time. The zero page list was created to initialize a predetermined number of demand-zero pages for future use by using idle, otherwise wasted cycles. Demand-zero pages are used for activating images and creating page tables. The dynamic system parameter ZERO_LIST_HI limits the maximum number of zero pages on the zero page list. Pages are zeroed and added to the zero page list for future use when the following conditions are met:

- The system is idle.
- Sufficient memory is available.
- The number of pages on the zero page list is less than the ZERO_LIST_HI value.

A.2.3 Image Slicing and the OpenVMS AXP Executive

The term **image slicing** refers to like parts of the OpenVMS AXP executive, drivers, and installed libraries that are grouped together so that they can be covered by a GHR and, therefore, loaded as a single unit. The term **executive** refers to those components that reside in system space. OpenVMS AXP uses GHRs for VMS executive images, drivers, and basic libraries (for example, LIBRTL and LIBOTS) by slicing modules and loading them into main images and image sections of like characteristics. Image slicing of the executive is the default on OpenVMS AXP and is controlled by the static system parameter LOAD_SYS_IMAGES.

The benefits of executive image slicing are as follows:

- A single TB entry can reference multiple pages, thus making it extremely unlikely that a TB miss will occur on those pages.
- More TB entries are available for use by the user and by other products running on the system.

To take full advantage of the ability to map multiple contiguous pages with a single TB entry using GHRs, OpenVMS AXP loads all nonpaged system code and data defined during startup into GHRs. Because the types and access characteristics of pages grouped together with a single TB entry must be the same, there is a TB entry for nonpaged code and another TB entry for nonpaged data.

These GHRs include nonpaged pages from executive images, drivers, and libraries installed with the /RESIDENT qualifier during the regular OpenVMS AXP startup procedure. (See Section A.2.5.3 for information about the /RESIDENT qualifier.) Note that drivers and shareable libraries, as well as all phases of VMS startup (including the execution of the SYSTARTUP_VMS.COM procedure) can be added to GHRs after system startup by setting the system parameter GH_RSRVPGCNT to allow enough space. The paged portion of the OpenVMS AXP executive can also be loaded nonpaged.

A.2.4 Image Slicing and User-Written Shareable Images and Libraries

The ability to use the same image-slicing technique of the OpenVMS AXP executive and to make full use of AXP granularity hints is available to layered products, independent products, and user code through the user-library slicing capability of OpenVMS AXP. The user code must be created as a main image or as a shareable image using the OpenVMS linker. (See Section A.2.5.2 for information about enabling user image slicing and installing shareable image libraries.)

Image slicing can be performed on main images and user-written shareable libraries. Also, only code in main and shareable images is sliced. Unlike the sliced OpenVMS AXP executive, all data from sliced user libraries remains private. Image slicing of user-written shareable libraries allows you to install optional software products, independent products, and user-shareable image libraries into granularity hint regions.

A.2.5 Why Use Image Slicing?

Installation of users' images provides benefits for user pages similar to the benefits provided to OpenVMS AXP executive, drivers, and data pages.

- Performance benefits similar to executive image slicing

  On OpenVMS AXP, main images and shareable libraries can be linked so that they can be installed with the /RESIDENT qualifier. The /RESIDENT qualifier causes nonpaged code image sections with like characteristics to be loaded together into a GHR in the system virtual address range and to be mapped by a single TB entry with a granularity hint. However, some patterns of use of shareable libraries might be data bound (that is, they might involve frequent accesses to copy-on-reference data sections). In such cases, improvement from installing with the /RESIDENT qualifier might be minimal.

- More efficient use of memory

  When you use the /RESIDENT qualifier to install code from frequently used main images and shareable libraries, the nonpaged code is grouped together in a GHR. Such grouping provides more efficient use of TB entries. It also results in more tightly packed code, thereby minimizing unused space between libraries and improving the efficiency of memory use.

A.2.5.1 Code Considerations

You should avoid certain code constructs in creating libraries that are installed with the /RESIDENT qualifier. Image slicing of main images and user-written shareable libraries is restricted to code sections with attributes EXE and NOWRT. Also, you can add all the routines that the main routine calls to a shareable library and install the shareable library, leaving only the main routine in the private image space.

A.2.5.2 How to Link

The /SHARE and /SECTION_BINDING qualifiers to the LINK command are required for enabling slicing of user-shareable libraries. The /SECTION_BINDING qualifier accepts two keywords—CODE and DATA. The CODE keyword tells the linker not to optimize calls between image sections for sliced libraries because assumptions of relative position of code image sections cannot be made at link time. Rather, the relative positions of code image sections are determined at load time.

The DATA keyword tells the linker to compress data image sections so that the data sections are contiguous to each other. To use the DATA keyword successfully, there can be no relative references between data image sections. For more information about the /SECTION_BINDING qualifier, see the *OpenVMS Linker Utility Manual.*

When using shareable libraries from an image (particularly shareable libraries installed resident), be sure to link the image with the shareable library and not with an equivalent object library for the shareable image. If you link the image with an equivalent object library and the shareable image, objects from the object library will be included in the image. As a result, the shareable library will not be used during run time. (The necessary routines were already pulled in from the object library during the link operation.)

A.2.5.3 Installing the Shareable Library

You should install a shareable user library as a resident image. Installing a shareable library for image slicing with the /RESIDENT qualifier to the INSTALL command implies installing with the following qualifiers:

- /HEADER_RESIDENT
- /OPEN
- /SHARE

For more information about the Install utility, the INSTALL command, and the /RESIDENT qualifier, see the *OpenVMS System Management Utilities Reference Manual.*

If you install the library during OpenVMS AXP startup, that is, in SYSTARTUP_VMS.COM, then the library is automatically included in the granularity hint region that contains the rest of the sliced executive, system drivers, and libraries. If you must install the library after OpenVMS AXP startup, set the system parameter GH_RSRVPGCNT to a value large enough to contain any user libraries or drivers that are installed after startup is complete.

A.2.5.4 System Parameters Related to Image Slicing

Table A–1 lists the system parameters that control image slicing.

Table A–1 System Parameters That Control Image Slicing

Parameter	Description
LOAD_SYS_IMAGES	Controls whether the executive is sliced during loading. Bit <0> enables loading of site-specific executive images into VMS$SYSTEM_IMAGES.DATA. Bit <1> enables slicing of the executive. Bit <2> enables release of unused portions of granularity hint regions (GHRs). The default value is 7 (bits <2:0> set).
GH_RSRVPGCNT	Specifies the number of unused pages within a GHR to be retained after startup. If bit <2> of LOAD_SYS_IMAGES is set, then pages used for executive and user image slicing are retained along with the number of pages specified by GH_RSRVPGCNT. The remaining pages in the GHR are returned for use by OpenVMS AXP.

(continued on next page)

Table A–1 (Cont.) System Parameters That Control Image Slicing

Parameter	Description
ITB_ENTRIES	Specifies the number of GHRs usable by OpenVMS AXP. The processor has four instruction translation buffer (ITB) entries for GHRs. OpenVMS AXP uses only a portion of one GHR for code sections, leaving the rest of the region for user library slicing. The current default for this parameter is 1. If a second GHR is necessary, this parameter can be increased to 4. If ITB_ENTRIES is set to 0, then the use of granularity hints for all types of images is disabled.

A.2.5.5 Other Tools and Utilities

Several other OpenVMS AXP system utilities, such as image accounting and user accounting information in ACCOUNTING, provide valuable information about your system and also allow you to set some system and user parameter values. The DCL commands SHOW and SET, along with AUTHORIZE and INSTALL, are some of the other utilities that are useful. In particular, INSTALL is useful for setting the most frequently accessed images as header resident so as to avoid several file system operations per activation and per run.

A.3 Performance Management

Performance management is key to ensuring that an OpenVMS AXP system functions efficiently. Managing the performance of an OpenVMS AXP system involves the following:

- Monitoring activity on the system
- Analyzing the resulting data
- Configuring your system to ensure optimal use of hardware and software resources for a given work load

This section provides some general background information about performance management, and discusses some common problems that might arise in the use of an OpenVMS AXP system. It describes the tools available to investigate and fix these problems. It is not intended to be comprehensive.

Performance management typically involves doing the following:

- Understanding and characterizing your work load and defining optimal or satisfactory conditions of operation
- Monitoring system activity on a regular basis to track trends in system activity and resource utilization
- Investigating and analyzing problems that degrade system performance
- Fixing performance problems

A.3.1 Characterizing the Work Load

In order to start down the correct path of investigation when a performance problem arises, you must understand your system's work load. This will help you evaluate the severity of the problem.

You must characterize your work load over a period of time by using the following information to define your operating environment:

- Number of users
- Kinds of tasks users perform on the system
- Load on the system as a function of time of day
- Peak hours of operation
- Utilization of key system resources during peak hours
- Frequently used applications

Several performance tools and utilities available on OpenVMS AXP can help you obtain this information.

A.3.2 Monitoring Performance

Efficient management of hardware and software resources requires regular monitoring of workload activity on your system.

One of your first important tasks for monitoring performance is to set up a process to monitor your system routinely. Use the following command procedures in SYS$EXAMPLES in sequence to begin this task:

1. SUBMON.COM—Starts MONITOR.COM as a detached process. You should invoke it from SYS$MANAGER:SYSTARTUP_VMS.COM.
2. MONITOR.COM—Creates a summary file from the recording file of the previous boot, and then starts recording for this boot. The default recording interval is 10 minutes.

These tools will allow you to obtain enough information about your system and its usage over a period of time to help you characterize your work load and identify a default, efficient, and satisfactory operating environment for your work load.

Once you have a good idea about the characteristics of your work load, you should ensure that you manage your resources efficiently. Four main categories of resources require monitoring:

- CPU
- Memory
- I/O subsystem
- Network

The key to efficient running of your system and to good performance is achieving a balance of use among the resources on your system.

You can use several tools and utilities, either alone or in combination, to monitor your OpenVMS AXP system and to track problems in specific areas. The most useful utilities are as follows:

- AUTOGEN
- Monitor utility (MONITOR)
- Accounting utility (ACCOUNTING)
- Install utility (INSTALL)
- DCL commands SET and SHOW

The scope of this section is to discuss performance monitoring using a set of utilities that come with the OpenVMS AXP operating system kit, with particular emphasis on using MONITOR. For detailed information about MONITOR, see the *OpenVMS System Management Utilities Reference Manual.*

A.3.2.1 Tuning System Parameters

Before your system can perform efficiently, it must be able to function well. This requires that the system provides the minimum necessary hardware and software resources for you to be able to use it. AUTOGEN is a tool that allows the configuration of software system parameters so that the system can boot and function. Though AUTOGEN is primarily a software configuration tool, it is useful as a performance tool in that it sets the values of system parameters to appropriate initial values for your system.

After installing a version of the OpenVMS AXP operating system, you should run AUTOGEN with feedback mode enabled to set the system parameter values to those appropriate for your system. If you have an idea about what some of the important parameter values for your workload environment should be, make sure you use MODPARAMS.DAT to specify these values for AUTOGEN to use. If you are setting up the system for the first time, then let AUTOGEN provide default values, which should be sufficient for most environments. If you run your work load for a period of time and then run AUTOGEN with feedback turned on, you will get better information about the values to which some of the parameters should be set. With this information, you can then use MODPARAMS.DAT to set the new parameter values and run AUTOGEN again. Note that you must reboot the system after running AUTOGEN in order for the changes to take effect. Refer to the *OpenVMS System Manager's Manual* for information about how to use AUTOGEN.

After a couple of iterations of using AUTOGEN with feedback, your system parameters will be tuned for efficient operation based on your workload environment. You are now ready to start routine monitoring of your system activity as a first step toward understanding and characterizing your workload requirements and to track resource utilization.

A.3.2.2 Monitoring System Activity

Important considerations about your system work load include the number of users, how active they are at different periods during a typical day, and what applications and images they run most frequently. At this stage, you should characterize your users and user activity at the highest level. You can do this in two ways:

- By using the image accounting feature of ACCOUNTING
- By using the MONITOR SYSTEM command

Image accounting data provides data about the users on the system. It also provides a list of images that are activated and some additional information, such as the amount of CPU time consumed, I/O operations, and page faults on a per-image or per-user basis. The MONITOR SYSTEM command gives you snapshots of the number of users on your system at each snapshot interval.

A.3.2.3 Monitoring the CPU Resource

Good system performance requires good CPU performance. On OpenVMS AXP systems, good CPU performance requires additional considerations such as the following:

- Minimal stalls caused by TB cache, instruction cache, or data cache misses
- Minimal instances of misaligned data

Fortunately, compilers bear most of this burden. Some of the previous sections discuss the performance features of the AXP hardware and of the OpenVMS AXP operating system. Users should exploit these features, as should applications running on your system.

Use the MONITOR MODES and MONITOR STATES commands to monitor CPU utilization and the various scheduling states of the processes on your system. MONITOR MODES breaks down system utilization by access mode. If your CPU is close to 100% utilization on the average over long monitoring periods, a CPU bottleneck is indicated. Consider the access mode in which the system spends most of its time. If it is kernel and interrupt state (not likely), then your CPU is executing primarily system code. Enter the MONITOR PROCESSES/TOPCPU command to look into which processes use most of the time. Then investigate what images these processes are running to narrow the problem to a specific area in the operating system. You can also use image accounting to look at the images that consume most of the CPU resource.

If the summary data from the MONITOR STATES command indicates that most of your processes are in the COM scheduling state, a CPU bottleneck is indicated. Your system and work load are compute bound, and you might have to redistribute your work load over different intervals of time or get additional CPUs to offset the CPU load.

A.3.2.4 Monitoring Memory

Like the CPU, the memory resource also plays a critical role in providing good system responsiveness and performance. The OpenVMS AXP memory management subsystem has a rich set of features that ensure equitable allocation of memory to needy processes and maintain a good balance between demand for memory and its supply. In addition, most OpenVMS AXP systems come configured with large physical memory. Such systems should provide a high-performance memory subsystem.

Note that there are some significant differences in the memory management subsystem on the OpenVMS AXP system as compared with that on an OpenVMS VAX system. Notable among these differences are granularity hint regions (GHRs) and the zero page list. For example, GHRs can alter the behavior or characteristics of the system considerably as compared with OpenVMS VAX systems. If you are used to an OpenVMS VAX system, be aware of this when you monitor the memory management subsystem and I/O activity, and analyze the results. Also, the paging rates you see might be less than what you are used to because the page sizes are larger and, therefore, more data is brought in a single page read I/O.

The most important system activities that indicate memory activity are page faults and swapping. Use the following commands to monitor memory activity:

- MONITOR PAGE—Provides statistics about the overall page fault rates and their breakdown into specific types of faults.
- MONITOR IO—Provides information about inswap rate.

- MONITOR STATES—Provides information about the number of processes in either outswapped states or involuntary memory-management-related wait states, such as FPG, PFW, or COLPG.

Monitor these activities and correlate them with your system performance metric (user response time, task elapsed time, or system throughput). With the adaptive pool management features (described in Section 5.4), the old rules of thumb about page faults being preferable to swapping are no longer valid. Rather, you should correlate each activity with your system performance metric and verify which activity most affects your work load. Depending on the type of work load, either activity (page faulting or swapping) might be the right choice and the system attempts to do the right thing based on the type of processes in your work load. Also, with adaptive pool management, you no longer need to manage and tune the individual SRP, IRP, and LRP lists. The user need only set the value of nonpaged pool to a reasonable value. With AUTOGEN feedback and adaptive pool management, the system provides optimally sized lookaside list packets based on system utilization.

A.3.2.5 Monitoring the I/O Subsystem

The I/O subsystem consists of the disk, tape, and user interface hardware and software subsystems, such as DECwindows Motif for OpenVMS AXP and terminals. Disk I/O plays a critical role in the overall responsiveness of your system, since it typically is the slowest component when compared with the CPU and memory resources. Disks are the primary means of secondary storage. Therefore, this section concentrates on disk I/O performance.

The MONITOR DISK/ITEM=ALL command allows you to obtain statistics on I/O rates to the disks on your system. It also indicates the average queue lengths for these disks. With this information, you can calculate the average response time for an I/O request for a given disk using Little's Law.

Little's Law states that the number of requests in a system is equal to the product of the throughput of that system and the average time a request spends in that system. This relationship is a fundamental principle in queueing theory and is widely applicable while requiring only weak assumptions about a queueing system. Applied to disk I/O requests on an OpenVMS AXP system, I/O throughput is provided by the I/O operation rate statistic. The average number of requests in the system is provided by the I/O request queue length statistic as a result of using the MONITOR DISK/ITEM=ALL command. Given these two considerations, you can calculate the average response time using the following expression:

$$average\ response\ time = \frac{average\ I/O\ request\ queue\ length}{average\ I/O\ operation\ rate}$$

If the average I/O rate to a disk is 20 I/O operations per second and the I/O queue length is 1.2 using the MONITOR DISK/ITEM=ALL command, then a good estimate of the average response time for an I/O to the device is 1.2/20 = 0.06 seconds, or 60 milliseconds (ms). On the other hand, if the I/O queue length is 0.3, then the average response time can be calculated to be 0.3/20 = 0.015 seconds, or 15 ms.

Disk I/O activity can be due either to system operations (for example, page read, inswap or outswap I/Os or file system maintenance) or to user applications. In most cases, the following MONITOR commands provide the statistics of interest in this area:

- MONITOR PAGE

- MONITOR IO
- MONITOR FCP
- MONITOR FILE_SYSTEM_CACHE

Also, network I/Os might occur if the network is running on your system while you are running DECwindows Motif for OpenVMS AXP. Using a RAM disk can reduce disk I/O activity. Specifically, DECram for OpenVMS is a disk device driver that allows a system manager to create pseudodisks (RAM disks) that reside in main memory for the purpose of improving I/O performance. These RAM disks can be accessed through the file system just as physical disks are accessed, but access to RAM disks is much faster.

A.3.2.6 Monitoring the Network

Networking is commonplace in most computing environments. In contrast to monitoring of resources on a single AXP system, the network resource is common to several systems that use it as a means of communication. The network subsystem consists of the hardware and software resources that comprise the computer network—primarily the communication medium such as the Ethernet local area network (LAN), the network adapters, and networking software such as DECnet for OpenVMS (DECnet). Network activity takes place mostly as a series of messages exchanged between two communicating processes on two different systems. These messages are transformed into packets on the communication medium. Several layered protocols enable the proper sending and receiving of messages between systems.

On each system, network activity uses some amount of CPU, memory, and I/O resources. Also, because the communication medium is usually shared among all systems connected to the network, such as an Ethernet LAN, you should monitor the network activity on your wire and characterize it in terms of heavy users, capacity, response time for packets on the wire, and so on.

MONITOR provides a DECNET class that you can use to track incoming and outgoing packets or network I/Os to and from a system. The Network Control Program (NCP) utility allows you to look at counters in the Data Link and Transport layers of DECnet and to obtain statistics about data transfer rates and problems with buffer availability. For example, line counters can provide a measure of throughput, collisions, and various error conditions associated with DECnet; circuit counters offer similar information about Ethernet circuits and about all users of those circuits, such as DECnet, VMScluster systems, and Local Area Transport (LAT). NCP also allows you to set certain network parameter values, notably the number and size of Transport layer transmit buffers, for efficient operation of that node in your network. Critical DECnet parameters for good performance are EXECUTOR PIPELINE QUOTA and LINE RECEIVE BUFFERS, as shown in the following table:

Parameter	Recommendation
EXECUTOR PIPELINE QUOTA[1]	Small value for all LANs. Consider increasing this value for slow, long-distance wide area networks (WANs).
LINE RECEIVE BUFFERS	Up to approximately 20. The default value is generally sufficient.

[1] The default value might be too high. See Section A.3.2.6.1 and Section A.3.2.6.2.

For details, refer to the *DECnet for OpenVMS Networking Manual* and the *DECnet for OpenVMS Guide to Networking*.

A.3.2.6.1 Problems with LAN-Based DECnet for OpenVMS Environments

Poor performance can occur in LAN-based Phase IV DECnet environments. It can take the following forms:

- Inconsistent variations in DECnet I/O rates
- Receiver overruns
- Significant amounts of explicit flow control

Typically, this condition affects AXP and VAX systems using DECnet. In particular, those systems using PATHWORKS to serve PCs and DECwindows client/server environments.

Poor adjustment of executor pipeline quota and buffer management parameters can cause this problem. In addition, further complications result from Ethernet adapter implementation variations.

How Is the Executor Pipeline Quota Involved?

The executor pipeline quota determines the maximum transmit window size. It represents the maximum number of packets that are transmitted before asking the receiver for an ACK (implicit flow control). You can define its value using NCP. The following equation describes the relationship between the pipeline quota and the transmit window size:

$$maximum\ transmit\ window\ size = \frac{executor\ pipeline\ quota}{executor\ buffer\ size}$$

The default executor buffer size is 576. The following equation shows the relationship between the initial transmit window size and the maximum transmit window size:

$$initial\ transmit\ window\ size = \frac{maximum\ transmit\ window\ size}{3} + 1$$

NSP flow control algorithms raise and lower the transmit window size between 1 and the maximum transmit window size.

A.3.2.6.2 How Are These Problems Resolved? You can optimize DECnet performance by tuning the number of transmitter buffers to mesh with the speed and window size of the Ethernet receiver.

Matching transmit buffer pipelines to receive pipeline depth and speed eliminates explicit flow control messages, eliminates dropped packets due to overruns, reduces “bursty” network behavior, and improves overall network efficiency.

Setting the DECnet executor pipeline quota in the range of 1728 to 4032 provides optimal performance for a wide range of configurations without any performance penalties. For AXP and VAX systems communicating exclusively with AXP and VAX systems or other fast CPUs and Ethernet adapters, set the executor pipeline quota to 4032, as shown in the following example:

```
$ MCR NCP
NCP>SET EXECUTOR PIPELINE QUOTA 4032
NCP>DEFINE EXECUTOR PIPELINE QUOTA 4032
NCP>EXIT
```

For AXP and VAX systems communicating exclusively with slow PCs or similarly slow CPUs and Ethernet adapters, set the executor pipeline quota to 1728.

Changing the value of the executor pipeline quota is dynamic, so that testing can be performed easily and safely on a running system. Logical links that already are established do not pick up the changes in the executor pipeline quota dynamically. However, changes do affect all newly established links.

B

I/O Subsystem Configuration Commands in SYSMAN

This appendix describes the I/O subsystem configuration support that is added to the System Management utility (SYSMAN) for OpenVMS AXP.

B.1 I/O Subsystem Configuration Support in SYSMAN

On OpenVMS AXP, SYSMAN is used to connect devices, load I/O device drivers, and display configuration information useful for debugging device drivers. I/O commands are in the System Generation utility (SYSGEN) on OpenVMS VAX.

Enter the following command to invoke SYSMAN:

```
$ SYSMAN :== $SYS$SYSTEM:SYSMAN
SYSMAN>
```

All SYSMAN commands that control and display the I/O configuration of an OpenVMS AXP computer must be preceded by "IO". For example, to configure a system automatically, enter the following command:

```
SYSMAN> IO AUTOCONFIGURE
```

This section contains a syntax description for the SYSMAN IO commands AUTOCONFIGURE, CONNECT, LOAD, SET PREFIX, SHOW BUS, SHOW DEVICE, and SHOW PREFIX.

IO AUTOCONFIGURE

Automatically identifies and configures all hardware devices attached to a system. The IO AUTOCONFIGURE command connects devices and loads their drivers.

You must have CMKRNL and SYSLCK privileges to use the IO AUTOCONFIGURE command.

Format

IO AUTOCONFIGURE

Parameters

None.

Qualifiers

/SELECT=(device-name)
Specifies the device type to be configured automatically. Use valid device names or mnemonics that indicate the devices to be included in the configuration. Wildcards must be explicitly specified.

The /SELECT and /EXCLUDE qualifiers are not mutually exclusive, as they are on OpenVMS VAX. Both qualifiers can be specified on the command line.

Table B–1 shows /SELECT qualifier examples.

Table B–1 /SELECT Qualifier Examples

Command	Configured Devices	Unconfigured Devices
/SELECT=P*	PKA,PKB,PIA	None
/SELECT=PK*	PKA,PKB	PIA
/SELECT=PKA*	PKA	PKB,PIA

/EXCLUDE=(device-name)

Specifies the device type that should not be configured automatically. Use valid device names or mnemonics that indicate the devices to be excluded from the configuration. Wildcards must be explicitly specified.

The /SELECT and /EXCLUDE qualifiers are not mutually exclusive, as they are on OpenVMS VAX. Both qualifiers can be specified on the command line.

/LOG

Controls whether the IO AUTOCONFIGURE command displays information about loaded devices.

Description

The IO AUTOCONFIGURE command identifies and configures all hardware devices attached to a system. You must have CMKRNL and SYSLCK privileges to use the IO AUTOCONFIGURE command.

Examples

1.
```
SYSMAN> IO AUTOCONFIGURE/EXCLUDE=DKA0
```

The command in this example autoconfigures all devices on the system except for DKA0.

IO AUTOCONFIGURE automatically configures all standard devices that are physically attached to the system, except for the network communications device.

2.
```
SYSMAN> IO AUTOCONFIGURE/LOG
```

The /LOG qualifier displays information about all the devices that AUTOCONFIGURE loads.

IO CONNECT

Connects a hardware device and loads its driver, if the driver is not already loaded.

You must have CMKRNL and SYSLCK privileges to use the IO CONNECT command.

Format

```
IO CONNECT device-name[:]
```

Parameters

device-name[:]
Specifies the name of the hardware device to be connected. It should be specified in the format *device-type controller unit-number*. For example, in the designation LPA0, LP is a line printer on controller A at unit number 0. If the /NOADAPTER qualifier is specified, the device is the software device to be loaded.

Qualifiers

/ADAPTER=tr-number
/NOADAPTER (default)
Specifies the nexus number of the adapter to which the specified device is connected. It is a nonnegative 32-bit integer. The /NOADAPTER qualifier indicates that the device is not associated with any particular hardware. The /NOADAPTER qualifier is compatible with the /DRIVER_NAME qualifier only.

/CSR=csr-address
Specifies the CSR address for the device being configured. This address must be specified in hexadecimal. You must prefix the CSR address with %X. The CSR address is a quadword value that is loaded into IDB$Q_CSR without any interpretation by SYSMAN. This address can be physical or virtual, depending on the specific device being connected:

- /CSR=%X3A0140120 for a physical address
- /CSR=%XFFFFFFFF807F8000 for a virtual address (the sign extension is required for AXP virtual addresses)

This qualifier is required if /ADAPTER=*tr-number* is specified.

/DRIVER_NAME=filespec
Specifies the name of the device driver to be loaded. If this qualifier is not specified, the default is obtained in the same manner as the SYSGEN default name. For example, if you want to load the SYS$ELDRIVER.EXE supplied by Digital, the prefix SYS$ must be present. Without the SYS$, SYSMAN looks for ELDRIVER.EXE in SYS$LOADABLE_IMAGES. This implementation separates the user device driver namespace from the device driver namespace supplied by Digital.

/LOG=(ALL,CRB,DDB,DPT,IDB,SB,UCB)
/NOLOG (default)

Controls whether SYSMAN displays the addresses of the specified control blocks. The default value for the /LOG qualifier is /LOG=ALL. If /LOG=UCB is specified, a message similar to the following is displayed:

```
%SYSMAN-I-IOADDRESS, the UCB is located at address 805AB000
```

The default is /NOLOG.

/MAX_UNITS=maximum-number-of-units

Specifies the maximum number of units the driver can support. The default is specified in the driver prologue table (DPT) of the driver. If the number is not specified in the DPT, the default is 8. This number must be greater than or equal to the number of units specified by /NUM_UNITS. This qualifier is optional.

/NUM_UNITS=number-of-units

Specifies the number of units to be created. The starting device number is the number specified in the device name parameter. For example, the first device in DKA0 is 0. Subsequent devices are numbered sequentially. The default is 1. This qualifier is optional.

/NUM_VEC=vector-count

Specifies the number of vectors for this device. The default vector count is 1. The /NUM_VEC qualifier is optional. This qualifier should be used only when using the /VECTOR_SPACING qualifier. When using the /NUM_VEC qualifier, you must also use the /VECTOR qualifier to supply the base vector.

/SYS_ID=number-of-remote-system

Indicates the SCS system ID of the remote system to which the device is to be connected. It is a 64-bit integer; you must specify the remote system number in hexadecimal. The default is the local system. This qualifier is optional.

/VECTOR=(vector-address,...)

Specifies the interrupt vectors for the device or lowest vector. This is either a byte offset into the SCB of the interrupt vector for directly vectored interrupts or a byte offset into the ADP vector table for indirectly vectored interrupts. The values must be longword aligned. To specify the vector addresses in octal or hexadecimal, prefix the addresses with %O or %X, respectively. This qualifier is required when /ADAPTER=*tr-number* or /NUM_VEC=*vector-count* is specified. Up to 64 vectors can be listed.

/VECTOR_SPACING=number-of-bytes-between-vectors

Specifies the spacing between vectors. Specify the amount as a multiple of 16 bytes. The default is 16. You must specify both the base vector with /VECTOR and the number of vectors with /NUM_VEC. This qualifier is optional.

Description

The IO CONNECT command connects a hardware device and loads its driver, if the driver is not already loaded. You must have CMKRNL and SYSLCK privileges to use the IO CONNECT command.

Examples

1.

```
SYSMAN> IO CONNECT DKA0:/DRIVER_NAME=SYS$DKDRIVER/CSR=%X80AD00-
/ADAPTER=4/NUM_VEC=3/VECTOR_SPACING=%X10/VECTOR=%XA20/LOG

%SYSMAN-I-IOADDRESS, the CRB is located at address 805AEC40
%SYSMAN-I-IOADDRESS, the DDB is located at address 805AA740
%SYSMAN-I-IOADDRESS, the DPT is located at address 80D2A000
%SYSMAN-I-IOADDRESS, the IDB is located at address 805AEE80
%SYSMAN-I-IOADDRESS, the SB is located at address 80417F80
%SYSMAN-I-IOADDRESS, the UCB is located at address 805B68C0
```

This command example connects device DKA0, loads driver SYS$DKDRIVER, and specifies the following:

Physical CSR address
Adapter number
Number of vectors
Spacing between vectors
Interrupt vector address

The /LOG qualifier displays the addresses of all control blocks.

2.

```
SYSMAN> IO CONNECT DKA0:/DRIVER_NAME=SYS$DKDRIVER/CSR=%X80AD00-
/ADAPTER=4/VECTOR=(%XA20,%XA30,%XA40)/LOG=(CRB,DPT,UCB)

%SYSMAN-I-IOADDRESS, the CRB is located at address 805AEC40
%SYSMAN-I-IOADDRESS, the DPT is located at address 80D2A000
%SYSMAN-I-IOADDRESS, the UCB is located at address 805B68C0
```

This command example connects device DKA0, loads driver SYS$DKDRIVER, and specifies the following:

Physical CSR address
Adapter number
Addresses for interrupt vectors

The /LOG qualifier displays the addresses of the channel request block (CRB), the driver prologue table (DPT), and the unit control block (UCB).

3.

```
SYSMAN> IO CONNECT FTA0:/DRIVER=SYS$FTDRIVER/NOADAPTER/LOG=(ALL)

%SYSMAN-I-IOADDRESS, the CRB is located at address 805AEC40
%SYSMAN-I-IOADDRESS, the DDB is located at address 805AA740
%SYSMAN-I-IOADDRESS, the DPT is located at address 80D2A000
%SYSMAN-I-IOADDRESS, the IDB is located at address 805AEE80
%SYSMAN-I-IOADDRESS, the SB is located at address 80417F80
%SYSMAN-I-IOADDRESS, the UCB is located at address 805B68C0
```

This command example connects pseudoterminal FTA0, loads driver SYS$FTDRIVER, and uses the /NOADAPTER qualifier to indicate that FTA0 is not an actual hardware device. The /LOG=ALL qualifier displays the addresses of all control blocks.

IO LOAD

Loads an I/O driver.

You must have CMKRNL and SYSLCK privileges to use the IO LOAD command.

Format

IO LOAD filespec

Parameters

filespec
Specifies the file name of the driver to be loaded. This parameter is required.

Qualifiers

/LOG=(ALL,DPT)
Controls whether SYSMAN displays information about drivers that have been loaded. The default value for the /LOG qualifier is /LOG=ALL. The driver prologue table (DPT) address is displayed when either /LOG=DPT or /LOG=ALL is specified.

Description

The IO LOAD command loads an I/O driver. You must have CMKRNL and SYSLCK privileges to use the IO LOAD command.

Example

```
SYSMAN> IO LOAD/LOG SYS$DKDRIVER
%SYSMAN-I-IOADDRESS, the DPT is located at address 80D5A000
```

This example loads device SYS$DKDRIVER and displays the address of the driver prologue table (DPT).

IO SET PREFIX

Sets the prefix list that is used to manufacture the IOGEN Configuration Building Module (ICBM) names.

Format

IO SET PREFIX=(icbm-prefix)

Parameters

icbm-prefix
Specifies ICBM prefixes. These prefixes are used by the IO AUTOCONFIGURE command to build ICBM image names.

Qualifiers

None.

Description

The IO SET PREFIX command sets the prefix list that is used to manufacture ICBM names.

Example

```
SYSMAN> IO SET PREFIX=(SYS$,PSI$,VME_)
```

This example specifies the prefix names used by IO AUTOCONFIGURE to build the ICBM names. The prefixes are SYS$, PSI$, and VME_.

IO SHOW BUS

On OpenVMS AXP systems, lists all the buses, node numbers, bus names, TR numbers, and base CSR addresses on the system. This display exists primarily for internal engineering support.

Format

IO SHOW BUS

Parameters

None.

Qualifiers

None.

Description

The IO SHOW BUS command lists all the buses, node numbers, bus names, TR numbers, and base CSR addresses. This display exists primarily for internal engineering support.

You must have CMKRNL privilege to use IO SHOW BUS.

Example

```
SYSMAN> IO SHOW BUS

 Bus          Node TR#  Name            Base CSR
LSB           0    1    EV3 4MB        FFFFFFFF86FA0000
LSB           6    1    MEM            FFFFFFFF86FC4000
LSB           7    1    MEM            FFFFFFFF86FCA000
LSB           8    1    IOP            FFFFFFFF86FD0000
    XZA XMI-SCSI  0    3    XZA-SCSI       0000008001880000
    XZA XMI-SCSI  1    3    XZA-SCSI       0000008001880000
    XZA XMI-SCSI  0    4    XZA-SCSI       0000008001900000
    XZA XMI-SCSI  1    4    XZA-SCSI       0000008001900000
  XMI           4    2    LAMB           0000008001A00000
    DEMNA         0    5    Generic XMI    0000008001E80000
    DEMNA         0    6    Generic XMI    0000008001F00000
```

This IO SHOW BUS example is from a DEC 7000 Model 600 AXP. Displays vary among different AXP systems. The indentation levels are deliberate in this display. They indicate the hierarchy of the adapter control blocks in the system. The column titles in the display have the following meanings:

Column Title	Meaning
Bus	Identity of the bus
Node	Index into the associated bus array (the bus slot)
TR#	Nexus number of the adapter to which the specified device is connected
Name	Name of the device
Base CSR	Base CSR address of the device

IO SHOW DEVICE

Displays information on device drivers loaded into the system, the devices connected to them, and their I/O databases. All addresses are in hexadecimal and are virtual.

Format

IO SHOW DEVICE

Parameters

None.

Qualifiers

None.

Description

The IO SHOW DEVICE command displays information on the device drivers loaded into the system, the devices connected to them, and their I/O databases.

The IO SHOW DEVICE command specifies that the following information be displayed about the specified device driver:

Driver	Name of the driver
Dev	Name of each device connected to the driver
DDB	Address of the device's device data block
CRB	Address of the device's channel request block
IDB	Address of the device's interrupt dispatch block
Unit	Number of each unit on the device
UCB	Address of each unit's unit control block

All addresses are in hexadecimal and are virtual.

Refer to the *OpenVMS System Manager's Manual* for additional information about SYSMAN.

Example

```
SYSMAN> IO SHOW DEVICE
```

The following is a sample display produced by the IO SHOW DEVICE command:

```
__Driver________Dev_DDB______CRB______IDB______Unit_UCB_____
SYS$FTDRIVER
                FTA 802CE930 802D1250 802D04C0
                                                 0 801C3710
SYS$EUDRIVER
                EUA 802D0D80 802D1330 802D0D10
                                                 0 801E35A0
SYS$DKDRIVER
                DKI 802D0FB0 802D0F40 802D0E60
                                                 0 801E2520
SYS$PKADRIVER
                PKI 802D1100 802D13A0 802D1090
                                                 0 801E1210
SYS$TTDRIVER
OPERATOR
NLDRIVER
```

SYS$TTDRIVER, OPERATOR, and NLDRIVER do not have devices associated with them.

IO SHOW PREFIX

Displays the current prefix list used in the manufacture of IOGEN Configuration Building Module (ICBM) names.

Format

IO SHOW PREFIX

Parameters

None.

Qualifiers

None.

Description

The IO SHOW PREFIX command displays the current prefix list on the console. This list is used by the IO AUTOCONFIGURE command to build ICBM names.

Example

```
SYSMAN> IO SHOW PREFIX
%SYSMAN-I-IOPREFIX, the current prefix list is: SYS$,PSI$,VME_
```

This command example shows the prefixes used by IO AUTOCONFIGURE to build ICBM names.

C
Additional Considerations

In your role of supporting new users on OpenVMS AXP systems, you might encounter questions about the following additional topics:

- The new Help Message utility. See Section C.1.
- An easy method to determine the hardware type, architecture type, and page size for the node on which OpenVMS is running. See Section C.2.
- The online Bookreader documentation provided on the OpenVMS AXP compact disc. See Section C.3.
- Unsupported DCL commands. See Section C.4.
- Differences in password generation display. See Section C.5.
- Default editor for the EDIT command is TPU rather than EDT. See Section C.6.
- The TECO editor is available. See Section C.7.
- Shareable images in the DEC C RTL. See Section C.8.
- Run-time libraries listed not included in this version of OpenVMS AXP. See Section C.9.
- Compatibility between the VAX VMS and OpenVMS AXP Mathematics Libraries. See Section C.10.

C.1 Help Message Utility

Help Message is a versatile new utility that lets you quickly access online descriptions of system messages from the DCL prompt on a character-cell terminal (including DECterm windows).

Help Message displays message descriptions from the latest OpenVMS messages documentation (the most recent version of the *VMS System Messages and Recovery Procedures Reference Manual* plus any subsequent releases). In addition, the Help Message database can optionally include other source files, such as user-supplied messages documentation.

The staff of most medium-sized to large data centers often includes help desk personnel who answer questions about the computing environment from general users and programmers. Typically the system manager is a consultant or technical backup to the help desk specialists. If you find yourself in this role, you may want to alert the help desk personnel, as well as general users and programmers on OpenVMS AXP systems, about the availability of Help Message.

See the *OpenVMS System Messages: Companion Guide for Help Message Users* and the *OpenVMS System Manager's Manual* for details about using Help Message.

C.2 Using F$GETSYI to Display Hardware Type, Architecture Type, and Page Size

You can use the F$GETSYI lexical function to determine the hardware type, architecture type, and page size for the node on which OpenVMS is running. This might be helpful if the computing resources available to your users includes AXP and VAX computers.

Use a DCL command procedure similar to the following:

```
$ ! File name: SHOW_ARCH.COM
$ !
$ ! Simple command procedure to display node hardware type,
$ ! architecture type, page size, and other basic information.
$ !
$ say = "write sys$output"
$ say " "
$ say "OpenVMS process with PID " + "''f$getjpi("","PID")'"
$ say " running at " + "''f$time()'" + "."
$ say " "
$ say "Executing on a " + "''f$getsyi("HW_NAME")'"
$ say " named " + "''f$getsyi("NODENAME")'" + "."
$ say " "
$ say "Architecture type is "  + "''f$getsyi("ARCH_TYPE")'"
$ say " and architecture name is " + "''f$getsyi("ARCH_NAME")'" + "."
$ say " "
$ say "Page size is "  + "''f$getsyi("PAGE_SIZE")'" + " bytes."
$ !
$ exit
```

On a VAX VMS Version 5.5 node, output from the procedure is similar to the following display:

```
OpenVMS process with PID 3FE00B0E
 running at 18-NOV-1993 17:22:37.92.

Executing on a VAX 6000-620
 named NODEXX.

Architecture type is 1
 and architecture name is VAX.

Page size is 512 bytes.
```

On an OpenVMS AXP Version 1.5 node, output from the procedure is similar to the following display:

```
OpenVMS process with PID 2FC00126
 running at 18-NOV-1993 17:43:19.37.

Executing on a DEC 4000 Model 610
 named SAMPLE.

Architecture type is 2
 and architecture name is Alpha.

Page size is 8192 bytes.
```

Note

For the F$GETSYI lexical function, the PAGE_SIZE, ARCH_NAME, and ARCH_TYPE arguments do not exist on VMS systems predating Version 5.5.

C.3 Online Documentation on Compact Disc

The OpenVMS Extended Documentation Set is included on the OpenVMS AXP Version 1.5 compact disc (CD) in DECW$BOOK format. Users with a workstation and DECwindows Motif installed can view the manuals with the Bookreader application. Refer to the *OpenVMS AXP Version 1.5 Compact Disc User's Guide* for a list of the manuals on the CD and information about enabling access to and reading the online documents.

C.4 Unsupported DCL Commands

The following DCL commands are not supported on OpenVMS AXP:

- MONITOR POOL
- SET FILE/UNLOCK
- UNLOCK

C.5 Password Generation

On OpenVMS AXP systems, the password generation algorithm allows for future use of generation databases for non-English passwords. Because of this, the password generation logic does not return information that is required to perform English word hyphenation. As a result, the SET PASSWORD command cannot display a hyphenated word list as it does on OpenVMS VAX systems. This change is permanent on OpenVMS AXP and is intended to accommodate possible future support for alternate-language password generation databases.

C.6 Default Editor for EDIT Command

The default editor for the EDIT command on OpenVMS AXP is TPU. The default editor on VAX VMS Version 5.*n* is EDT. When users enter the EDIT command, the Extensible Versatile Editor (EVE) is invoked rather than the EDT editor. The EDT editor is still included with OpenVMS AXP.

If your users prefer to continue using the EDT editor, have them define the following symbol interactively or in their login command procedure:

```
$ EDIT :== EDIT/EDT
```

This symbol overrides the default and causes the EDIT command to use the EDT editor instead of EVE.

For any DCL command procedure that relies on EDT, verify that the /EDT qualifier is present on EDIT commands. Any procedure that uses the EDIT command without the /EDT qualifier will fail because this verb now invokes TPU with the TPU$SECTION section file. (By default, this section file is EVE.)

Note that the default editor for the Mail utility (MAIL) also has been changed to the DECTPU-based EVE editor rather than EDT. By entering the MAIL command SET EDITOR, you can specify that a different editor be invoked instead of the DECTPU editor. For example, to select the EDT editor, issue the MAIL command SET EDITOR EDT. The EDT editor remains your default MAIL editor (even if you log out of the system and log back in) until you enter the SET EDITOR TPU command.

Users also can define the logical name MAIL$EDIT to be a command file before entering MAIL. When they issue any MAIL command that invokes an editor, the command file will be called to perform the edit. In the command file, you can also invoke other utilities, such as the spellchecker, and you can specify any function that can be done in a command file.

If desired, another option is to override the selected MAIL editor temporarily by defining MAIL$EDIT to be the string CALLABLE_ with the name of the desired editor appended. For example, to use callable EDT rather than callable EVE, users can type the following command:

```
$ DEFINE MAIL$EDIT CALLABLE_EDT
```

In EVE, you can select an EDT-like keypad by defining the OpenVMS AXP logical name EVE$KEYPAD to be EDT. See the General Release Notes chapter of the *OpenVMS AXP Version 1.5 Release Notes* for details about how to do this at either the process level or the system level. You can find an example of how to define EVE$KEYPAD at the system level in the file SYS$STARTUP:SYLOGICALS.TEMPLATE.

The general release notes chapter of the *OpenVMS AXP Version 1.5 Release Notes* also contains a complete description of the EDIT command, DECTPU, and EVE.

See the *Guide to the DEC Text Processing Utility* and the *OpenVMS User's Manual* for more information about DECTPU and EVE.

C.7 TECO Editor is Available

The TECO editor is included in OpenVMS AXP. Invoke the TECO editor with the EDIT/TECO command as described in the *OpenVMS DCL Dictionary* or with more traditional access methods. For information about the use of TECO, see the *PDP–11 TECO User's Guide.*

C.8 Shareable Images in the DEC C RTL for OpenVMS AXP

If you answer problem reports submitted by programmers who are coding C applications on OpenVMS AXP systems, you might receive questions about the shareable images in the DEC C Run-Time Library (RTL).

On AXP systems, the DEC C RTL does not provide the VAXCRTL.EXE or VAXCRTLG.EXE shareable images. Instead, the image DECC$SHR.EXE (which resides in IMAGELIB) must be used. This image contains all DEC C RTL functions and data and has an OpenVMS conformant namespace (all external names are prefixed with DECC$). To use this image, all DEC C RTL references must be prefixed with DECC$ so that the proper code in DECC$SHR.EXE is accessed. This prefixing occurs as the default action of the compiler.

Note that on VAX systems you use an option file when using VAXCRTL.EXE. On AXP systems, you do not use an option file when using DECC$SHR.EXE because it is in SYS$LIBRARY:IMAGELIB.OLB.

See the *DEC C Run-Time Library Reference Manual for OpenVMS Systems* and the *DEC C Reference Supplement for OpenVMS AXP Systems* for more information.

C.9 Run-Time Libraries

Table C–1 lists the run-time libraries that are not included in OpenVMS AXP Version 1.5.

Table C–1 Run-Time Libraries Not Included in OpenVMS AXP Version 1.5

ADARTL	Ada is not supported.
DBGSSISHR	DEBUG item is not required on OpenVMS AXP.
DNSRTL, DNSSHARE, and DTI$SHARE	DNS is not supported.
EPM$SRVSHR	DECtrace is not supported.
VBLAS1RTL	OpenVMS VAX vector programs are not supported.
VMTHRTL	OpenVMS VAX vector programs are not supported.

Most run-time libraries that were available in VMS Version 5.4 are available in OpenVMS AXP Version 1.5. The VMS Version 5.4 libraries that are not available are either not being ported to OpenVMS AXP or are planned for a later release of OpenVMS AXP.

For example, the vector math libraries VBLAS1RTL and VMTHRTL are not available in OpenVMS AXP because there is no support on OpenVMS AXP for programs that use the VAX VMS vector instructions.

C.10 Compatibility Between the VAX VMS and OpenVMS AXP Mathematics Libraries

Mathematical applications using the standard OpenVMS call interface to the OpenVMS Run-Time Mathematics (MTH$) Library need not change their calls to MTH$ routines when migrating to an OpenVMS AXP system. Jacket routines are provided that map MTH$ routines to their math$ counterparts in the Digital Portable Mathematics Library (DPML) for OpenVMS AXP. However, there is no support in the DPML for calls made to JSB entry points and vector routines. Please note that DPML routines are different from those in the OpenVMS Run-Time Mathematics (MTH$) Library. You should expect to see small differences in the precision of the mathematical results.

If one of your goals is to maintain compatibility with future libraries and to create portable mathematical applications, Digital recommends that you use the DPML routines available through the high-level language of your choice (for example, Fortran and C) rather than using the call interface. Significantly higher performance and accuracy are also available with DPML routines.

See the *DPML, Digital Portable Mathematics Library* manual for more information about DPML.

Index

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825) and press 2 for technical assistance.

Electronic Orders

If you wish to place an order through your account at the Electronic Store, dial 800-234-1998, using a modem set to 2400- or 9600-baud. You must be using a VT terminal or terminal emulator set at 8 bits, no parity. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825) and ask for an Electronic Store specialist.

Telephone and Direct Mail Orders

From	**Call**	**Write**
U.S.A.	DECdirect Phone: 800-DIGITAL (800-344-4825) FAX: (603) 884-5597	Digital Equipment Corporation P.O. Box CS2008 Nashua, NH 03061
Puerto Rico	Phone: (809) 781-0505 FAX: (809) 749-8377	Digital Equipment Caribbean, Inc. 3 Digital Plaza, 1st Street Suite 200 Metro Office Park San Juan, Puerto Rico 00920
Canada	Phone: 800-267-6215 FAX: (613) 592-1946	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: DECdirect Sales
International	————	Local Digital subsidiary or approved distributor
Internal Orders[1] (for software documentation)	DTN: 241-3023 (508) 874-3023	Software Supply Business (SSB) Digital Equipment Corporation 1 Digital Drive Westminster, MA 01473
Internal Orders (for hardware documentation)	DTN: 234-4325 (508) 351-4325 FAX: (508) 351-4467	Publishing & Circulation Services Digital Equipment Corporation NR02-2 444 Whitney Street Northboro, MA 01532

[1] Call to request an Internal Software Order Form (EN–01740–07).

Reader's Comments

A Comparison of
System Management on
OpenVMS AXP and OpenVMS VAX
AA–PV71A–TE

Your comments and suggestions help us improve the quality of our publications.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (product works as manual says)	□	□	□	□
Completeness (enough information)	□	□	□	□
Clarity (easy to understand)	□	□	□	□
Organization (structure of subject matter)	□	□	□	□
Figures (useful)	□	□	□	□
Examples (useful)	□	□	□	□
Index (ability to find topic)	□	□	□	□
Page layout (easy to find information)	□	□	□	□

I would like to see more/less

What I like best about this manual is

What I like least about this manual is

I found the following errors in this manual:

Page Description

Additional comments or suggestions to improve this manual:

For software manuals, please indicate which version of the software you are using:

Name/Title Dept.
Company Date
Mailing Address
Phone

--- Do Not Tear - Fold Here and Tape ---

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OpenVMS Documentation
110 SPIT BROOK ROAD ZKO3–4/U08
NASHUA, NH 03062–2642

--- Do Not Tear - Fold Here ---