Migrating an Environment from OpenVMS VAX to OpenVMS Alpha

Order Number: AA–QSBLA–TE

December 1995

This manual describes how to migrate a computing environment from an OpenVMS VAX system to an OpenVMS Alpha system or a Mixed-Architecture Cluster

Revision/Update Information: This is a new manual.

Software Version: OpenVMS Alpha Version 7.0
OpenVMS VAX Version 7.0

Digital Equipment Corporation
Maynard, Massachusetts

December 1995

ZK6460

This document is available on CD–ROM.

Contents

8 Network Management Tasks

9 Interoperability of OpenVMS VAX and OpenVMS Alpha

10 Guidelines for Developing Applications for Mixed-Architecture VMScluster Systems

Preface

Migrating an Environment from OpenVMS VAX to OpenVMS Alpha is designed to assist developers in moving a computing environment from an OpenVMS VAX system to an OpenVMS Alpha system or a Mixed-Architecture Cluster. The manual consists of the following chapters:

- Chapter 1 describes the main similarities and differences between VAX and Alpha systems.
- Chapter 2 contains system management considerations that might be pertinent to your job of supporting OpenVMS general users and programmers working on Alpha computers.
- Chapter 3 explains why there are differences in OpenVMS system management on Alpha and VAX computers.
- Chapter 4 describes how OpenVMS system management setup tasks are similar or different on Alpha and VAX computers.
- Chapter 5 describes how OpenVMS system management maintenance tasks are similar or different on Alpha and VAX computers.
- Chapter 6 describes how OpenVMS system management security tasks are similar or different on Alpha and VAX computers.
- Chapter 7 describes how the OpenVMS system management tasks that are designed to optimize performance are similar or different on Alpha and VAX computers.
- Chapter 8 describes how DECnet network management tasks are similar or different on Alpha and VAX computers.
- Chapter 9 describes how OpenVMS VAX and OpenVMS Alpha systems can work together in networks and VMSclusters.

Intended Audience

This manual is intended for experienced software engineers and system or VMScluster managers responsible for moving a computing environment from OpenVMS VAX to OpenVMS Alpha.

Related Documents

This manual is part of a set of manuals that describes various aspects of migrating from OpenVMS VAX to OpenVMS Alpha systems. The other manuals in this set are as follows:

- *Migrating an Application from OpenVMS VAX to OpenVMS Alpha* describes how to create an OpenVMS Alpha version of an OpenVMS VAX application. It provides an overview of the VAX to Alpha migration process and information to help you plan a migration. It discusses the decisions you must make in planning a migration and the ways to get the information you need to make those decisions. In addition, it describes the migration methods available so that you can estimate the amount of work required for each method and select the method best suited to a given application.

- *Porting VAX MACRO Code from OpenVMS VAX to OpenVMS Alpha* describes how to port VAX MACRO code to an Alpha system using the MACRO–32 compiler for OpenVMS Alpha. It describes the features of the compiler, presents a methodology for porting VAX MACRO code, identifies nonportable coding practices, and recommends alternatives to such practices. The manual also provides a reference section with detailed descriptions of the compiler's qualifiers, directives, and built-ins, and the system macros created for porting to Alpha systems.

In addition, the *DECmigrate for OpenVMS AXP Systems Translating Images* manual describes the VAX Environment Software Translator (VEST) utility. This manual is distributed with the optional layered product, DECmigrate for OpenVMS Alpha, which supports the migration of OpenVMS VAX applications to OpenVMS Alpha systems. The manual describes how to use VEST to convert most user-mode VAX images to translated images that can run on Alpha systems; how to improve the run-time performance of translated images; how to use VEST to trace Alpha incompatibilities in an VAX image back to the original source files; and how to use VEST to support compatibility among native and translated run-time libraries. The manual also includes complete VEST command reference information.

For additional information on OpenVMS products and services, access the Digital OpenVMS World Wide Web site. Use the following URL:

`http://www.openvms.digital.com`

Reader's Comments

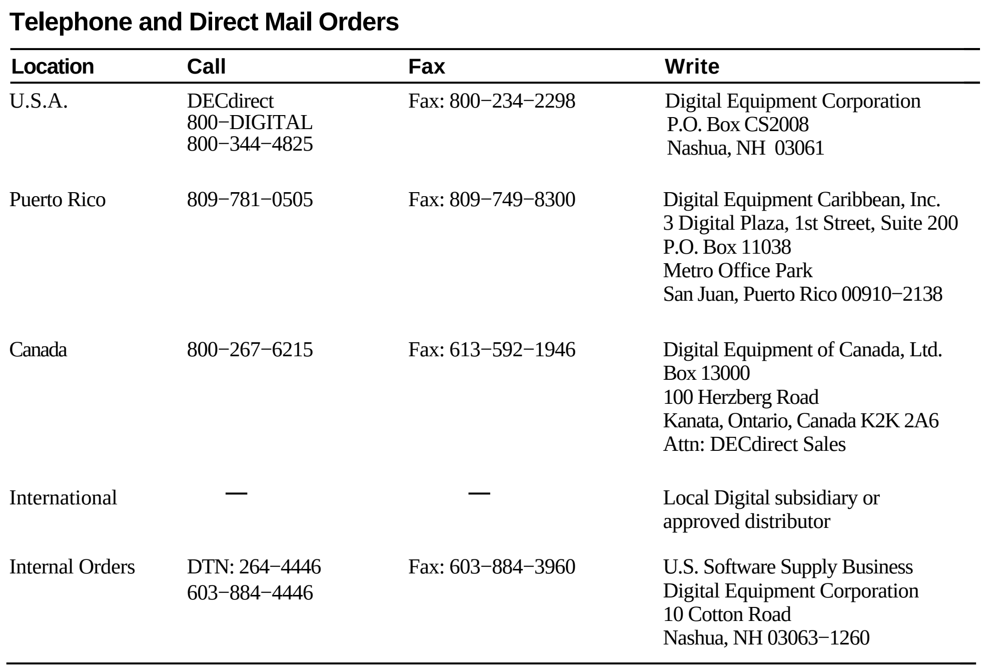
Digital welcomes your comments on this manual.

Print or edit the online form SYS$HELP:OPENVMSDOC_COMMENTS.TXT and send us your comments by:

Internet	**openvmsdoc@zko.mts.dec.com**
Fax	603 881-0120, Attention: OpenVMS Documentation, ZK03-4/U08
Mail	OpenVMS Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How To Order Additional Documentation

Use the following table to order additional documentation or information. If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825).



Telephone and Direct Mail Orders

Location	Call	Fax	Write
U.S.A.	DECdirect 800−DIGITAL 800−344−4825	Fax: 800−234−2298	Digital Equipment Corporation P.O. Box CS2008 Nashua, NH 03061
Puerto Rico	809−781−0505	Fax: 809−749−8300	Digital Equipment Caribbean, Inc. 3 Digital Plaza, 1st Street, Suite 200 P.O. Box 11038 Metro Office Park San Juan, Puerto Rico 00910−2138
Canada	800−267−6215	Fax: 613−592−1946	Digital Equipment of Canada, Ltd. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: DECdirect Sales
International	—	—	Local Digital subsidiary or approved distributor
Internal Orders	DTN: 264−4446 603−884−4446	Fax: 603−884−3960	U.S. Software Supply Business Digital Equipment Corporation 10 Cotton Road Nashua, NH 03063−1260

ZK−7654A−GE

Conventions

The name of the OpenVMS AXP operating system has been changed to OpenVMS Alpha. Any references to OpenVMS AXP or AXP are synonymous with OpenVMS Alpha or Alpha.

In this manual, every use of DECwindows and DECwindows Motif refers to DECwindows Motif for OpenVMS 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.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
. . .	Horizontal ellipsis points in examples indicate one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	Vertical ellipsis points indicate 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 command format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In command 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 command format descriptions, braces indicate 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 indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *device-name* contains up to five alphanumeric characters).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1 Overview

This chapter describes:

- Lineage of OpenVMS Alpha
- Similarities and differences at the DCL level
- Similarities and differences between OpenVMS VAX and OpenVMS Alpha in the end-user, system management, and programming environments

1.1 Introduction

The purpose of this manual is to provide OpenVMS VAX users with the information they need to assess the impact of adding one or more OpenVMS Alpha systems to their computing environments. The manual focuses on the similarities and differences between the OpenVMS VAX and OpenVMS Alpha operating systems.

OpenVMS Alpha runs on Digital's Alpha computers, which are reduced instruction set computers (RISC). OpenVMS Alpha systems provide as much compatibility as possible with OpenVMS VAX systems without compromising the advantages offered by the Alpha architecture. OpenVMS VAX customers can use the RISC technology of OpenVMS Alpha with little change in their computing environment.

Experienced OpenVMS VAX system managers will find their knowledge and most of their practices transferable to the OpenVMS Alpha environment. Many OpenVMS Alpha system managers compare the degree of change to that introduced by the change from VAX VMS Version 4.*n* to VAX VMS Version 5.0.

Many Digital customers use OpenVMS VAX to run their applications that require high standards of availability, scalability, and data integrity. They depend on its reliability, robust engineering, and leadership features, such as VAXcluster systems. Digital anticipates that many of its customers will add OpenVMS Alpha systems to their computing environments. Digital recognizes that this is an evolutionary process and will differ from customer to customer. Digital is committed to providing a clear coexistence environment and migration path between OpenVMS VAX and OpenVMS Alpha systems.

1.2 Lineage of OpenVMS Alpha

In 1978, Digital Equipment Corporation released Version 1.0 of the VMS operating system. Each new release since Version 1.0 represents a balance between compatibility with earlier releases and the introduction of new features and new technology that enable users to do their work more cost-effectively. A further development in this evolution was the introduction of OpenVMS AXP in November 1992.

Since the release of OpenVMS AXP Version 1.0, which was based on VMS Version 5.4–2, additional OpenVMS VAX features have been added to each new release. OpenVMS AXP Version 1.0 was followed by OpenVMS AXP Version 1.5. The next release of OpenVMS AXP was numbered Version 6.1 to acknowledge its functional equivalence with OpenVMS VAX Version 6.1. For Version 6.2, Digital changed the name of OpenVMS AXP to OpenVMS Alpha.

For the purposes of this manual, functional equivalence between OpenVMS VAX and OpenVMS Alpha is defined as the same type of functionality with possible slight variations in the implementation or the user interface.

When OpenVMS VAX Version 6.1 and OpenVMS AXP Version 6.1 were released, some exceptions to functional equivalence existed. Most of these exceptions were eliminated with Version 6.2.

1.3 End-User's Environment

The end-user's environment on OpenVMS Alpha is virtually the same as that on OpenVMS VAX, as described in the following sections.

1.3.1 DIGITAL Command Language (DCL)

The DIGITAL Command Language (DCL), the standard user interface to OpenVMS, remains essentially unchanged with OpenVMS Alpha. All commands and qualifiers available on OpenVMS VAX are also available on OpenVMS Alpha, except for a few, as shown in Table 2–1. In addition, a few qualifiers are available only on OpenVMS Alpha, as shown in Table 2–1.

1.3.2 DCL Help

DCL help is available on OpenVMS Alpha. Most of the DCL help text is common to both OpenVMS Alpha and OpenVMS VAX systems. For a few topics, information that is specific to one system is included in the display for both systems. System-specific information is identified by the phases "On VAX" and "On Alpha."

1.3.3 DCL Command Procedures

DCL command procedures that use commands, qualifiers, and lexical functions available on OpenVMS VAX continue to work on OpenVMS Alpha systems without change, except for command procedures that contain those few qualifiers not available on OpenVMS Alpha.

1.3.4 Databases

Standard databases, such as Oracle Rdb, function the same on OpenVMS VAX and OpenVMS Alpha systems. Most third-party databases that are available for OpenVMS VAX are also available for OpenVMS Alpha.

1.3.5 DECforms

The DECforms interface is unchanged. Applications that use the predecessor to DECforms, TDMS, can be moved to OpenVMS Alpha with the aid of a TDMS emulator or a TDMS converter. The VAX TDMS Emulator for OpenVMS and the VAX TDMS to DECforms Converter for OpenVMS VAX are produced by Praxa Limited of Melbourne, Australia and available through Digital.

1.3.6 DECwindows Motif

DECwindows Motif for OpenVMS VAX and DECwindows Motif for OpenVMS Alpha contain virtually identical functionality. The X11R5 display server that ships with OpenVMS Alpha does, however, contain additional extensions (for example, Shape, which gives the ability to create non-rectangular windows, and scalable fonts) that are not available on the OpenVMS VAX platform.

For more information about these X11R5 features and extensions, see *Managing DECwindows Motif for OpenVMS Systems*.

1.3.7 Editors and Formatter

The EVE and EDT editors are unchanged. EVE is the default editor for OpenVMS VAX and for OpenVMS Alpha. EDT was the default editor for OpenVMS VAX prior to Version 6.0.

The TECO editor and the DSR formatter are also provided with OpenVMS Alpha. They, too, are unchanged.

1.3.8 Help Message Utility

The Help Message utility is available on both OpenVMS VAX and OpenVMS Alpha systems. It enables users to access online descriptions of system messages on a character-cell terminal (including DECterm windows).

Help Message operates through a DCL interface that accesses message descriptions in a text file. This text file is derived from the latest version of the OpenVMS system messages documentation and, optionally, from other source files, including user-supplied message documentation. For more information about Help Message, see *OpenVMS System Messages: Companion Guide for Help Message Users*.

1.3.9 Password Generator

Both OpenVMS VAX and OpenVMS Alpha have random password generators, which can be used in place of passwords created by users. However, the random password generator on OpenVMS Alpha systems has a new mechanism for generating passwords. This new mechanism produces nonsense passwords such as *dramnock*, *inchworn*, *mycousia*, and *pestrals* that seem "natural" to users. It makes possible the future development of language-specific random passwords. The list of possible passwords presented by an OpenVMS Alpha system for selection is not hyphenated.

1.4 System Manager's Environment

Most of the OpenVMS VAX system management utilities, command formats, and tasks are identical in the OpenVMS Alpha environment.

Some components that were available only on OpenVMS VAX Version 6.1 or only on OpenVMS AXP Version 6.1 are now available on both systems, as shown in Table 1–1.

Table 1–1 Components That Achieved Functional Equivalence in OpenVMS Version 6.2

Component	Previously Available
Cluster event notification system services	Alpha only
DECamds Data Analyzer	VAX only
Dynamic device recognition	Alpha only
Full name support, including support by DECdtm	VAX only
LAT selected features	Alpha only
New proxy service and new security server	VAX only
RECALL command, additional support	Alpha only
SCSI–2 device support for Tagged Command Queuing	VAX only

However, you must consider some system management differences to properly set up, maintain, secure, and optimize OpenVMS Alpha systems and to establish proper network connections. These differences fall into two categories:

- Implementation differences for common components
- System management components not available on both OpenVMS VAX and OpenVMS Alpha

The differences are briefly described in this section and in greater detail in *A Comparison of System Management on OpenVMS AXP and OpenVMS VAX* and in the *OpenVMS System Manager's Manual*.

1.4.1 Common Components With Implementation Differences

The components described in this section exist on both OpenVMS VAX and OpenVMS Alpha systems, but the implementations on VAX and Alpha differ, or the OpenVMS VAX implementation differs from earlier versions of OpenVMS VAX.

1.4.1.1 Disk Quotas

You might need to increase disk quota on disks that store translated OpenVMS VAX images and native OpenVMS Alpha images. Translated images are OpenVMS Alpha executable images produced by the VAX Environment Software Translator (VEST), the major component of DECmigrate (described in *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*). The reasons why increased disk quotas are usually needed are listed in Table 1–2.

Table 1–2 Larger Disk Quotas Needed on OpenVMS Alpha

Condition	Explanation
Translated images	Require more disk space because each image includes both Alpha code and the original VAX code.
Native OpenVMS Alpha images	Require more disk space because RISC images typically contain more instructions and code to establish the linkage between procedure calls. The default values for related memory quotas have been adjusted on OpenVMS Alpha systems.

1.4.1.2 I/O Subsystem Configuration Commands

On OpenVMS VAX systems, the System Generation utility (SYSGEN) is used to configure the I/O subsystem. On OpenVMS Alpha systems, SYSGEN is used for some configuration tasks, and the System Management utility (SYSMAN) and the AUTOGEN command procedure are used for others, as shown in Table 1–3.

Table 1–3 A Comparison of I/O Subsystem Configurations

	OpenVMS VAX	OpenVMS Alpha
SYSGEN	Modify system parameters[1] Load page and swap files Create additional page files Create additional swap files Load device drivers	Modify system parameters[1] Load page and swap files Create additional page files Create additional swap files
SYSMAN	Not used	Load device drivers

[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.

You must modify OpenVMS VAX command procedures that use commands such as SYSGEN AUTOCONFIGURE ALL, if they are copied to OpenVMS Alpha systems as part of your migration effort; use the SYSMAN command IO AUTOCONFIGURE instead.

1.4.1.3 Menu-Driven Maintenance Procedure

A menu-driven maintenance procedure to replace the standalone BACKUP utility was introduced with the OpenVMS AXP Version 6.1 distribution CD–ROM. It was also provided on the OpenVMS VAX Version 6.1 distribution CD–ROM as an alternative to the standalone BACKUP utility (which is still currently supported on OpenVMS VAX systems).

The new procedure enables you to enter a DCL environment, from which you can perform backup and restore operations on the system disk (instead of using standalone BACKUP).

For more detailed information about using the menu-driven procedure, see the following documentation:

- On OpenVMS VAX, see the *OpenVMS System Manager's Manual*
- On OpenVMS Alpha, see the *OpenVMS Alpha Version 6.2 Upgrade and Installation Manual*

1.4.1.4 MONITOR POOL Command

The adaptive pool management feature of OpenVMS VAX and OpenVMS Alpha has made the DCL command MONITOR POOL obsolete. Adaptive pool management automatically manages the creation and sizing of nonpaged pool lookaside lists.

You can obtain information about nonpaged and paged dynamic pool on OpenVMS VAX through the use of the DCL command SHOW MEMORY and the System Dump Analyzer (SDA) command SHOW POOL. You can obtain the same information on OpenVMS Alpha with the SDA command SHOW POOL and its qualifiers /RING_BUFFER and /STATISTICS.

1.4.1.5 Name Changes for Files Supplied with the Operating Systems

The name changes for certain files supplied with the operating system are shown in Table 1–4.

Table 1–4 Operating System File Name Changes

OpenVMS VAX Version 5.5	OpenVMS VAX Version 6.0 and later	OpenVMS AXP Version 1.0 and later
SYSTARTUP_V5.COM	SYSTARTUP_VMS.COM	SYSTARTUP_VMS.COM
VAXVMSSYS.PAR	VAXVMSSYS.PAR	ALPHAVMSSYS.PAR

1.4.1.6 Page Size

OpenVMS VAX and OpenVMS Alpha systems allocate and deallocate memory for processes in units called **pages**. On OpenVMS VAX systems, a page is 512 bytes. On OpenVMS Alpha systems, a page is one of four values: 8 KB (8192 bytes), 16 KB, 32 KB, or 64 KB. Each OpenVMS Alpha system implements only one of the four page sizes; the initial set of Alpha computers use an 8KB page.

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

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

In your OpenVMS Alpha 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 Alpha system, 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 Alpha computer in use (8KB, 16KB, 32KB, or 64KB pages). In general, OpenVMS Alpha utilities display CPU-specific page values when the data represents physical memory.

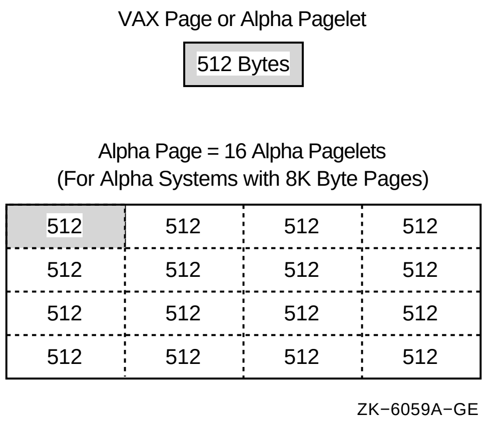
Internally, for the purposes of memory allocation, deletion, and protection, OpenVMS Alpha rounds up (if necessary) the value you supply in pagelets to a number of CPU-specific pages.

The use of pagelets provides compatibility for OpenVMS VAX system managers and application programmers who are accustomed to thinking about memory values in 512-byte units. In a VMScluster system with OpenVMS VAX and OpenVMS Alpha nodes, it is helpful to know that a VAX page and an Alpha 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 Alpha.

OpenVMS Alpha 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 information that each VAX page value and each Alpha pagelet value equal a common 512-byte quantity.

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

Figure 1–1 Comparison of VAX and Alpha Page Size


VAX Page or Alpha Pagelet
512 Bytes
Alpha Page = 16 Alpha Pagelets
(For Alpha Systems with 8K Byte Pages)
512
ZK−6059A−GE

1.4.1.7 Security

On Alpha systems, OpenVMS contains all the security features of OpenVMS VAX with the exception of DECnet connection auditing. OpenVMS VAX Version 6.0 was successfully evaluated at the C2 level in September 1993. A Security RAMP (Rating Maintenance Phase) is currently underway for OpenVMS VAX Version 6.1, and the C2 rating is expected in the very near future. A Security RAMP is also currently underway for OpenVMS AXP Version 6.1. OpenVMS VAX Version 6.2 and OpenVMS Alpha Version 6.2, and all future releases, will also be entered in the RAMP process in order to maintain the C2 Security rating.

1.4.1.8 VMScluster Systems

A VMScluster system can consist entirely of OpenVMS Alpha nodes or a combination of one or more OpenVMS VAX nodes and one or more OpenVMS Alpha nodes. The system management of a VMScluster system is essentially the same as that of a VAXcluster system.

For more information on VMScluster systems, see Section 9.4.

1.4.2 System Management Features Not Available on Both Systems

The features described in this section are not available on both OpenVMS VAX and OpenVMS Alpha systems. Components that exist only on OpenVMS VAX or only on OpenVMS Alpha are either planned, under investigation, or not planned for the other system, as shown in Table 1–5.

Table 1–5 System Management Features Not Available on Both Systems

Feature	On VAX or Alpha	Status
Card reader and input symbiont	VAX	Not planned for Alpha
DECevent utility	Alpha	Under investigation for VAX
DECnet connection auditing	VAX	Planned for Alpha
DECnet DDCMP support	VAX	Not planned for Alpha
DECnet host-based routing	VAX	Planned for Alpha
Dump file off the system disk	VAX	Planned for Alpha
Installation of the operating system with PCSI	Alpha	Planned for VAX
MSCP dynamic load balancing	VAX	Planned for Alpha
Patch utility	VAX	Limited functionality under investigation for Alpha
SCSI VMSclusters	Alpha	Under investigation for VAX (see Section 9.4)
Selected optional software products	VAX or Alpha	See Section 1.4.2.4
Shadowing dump file failover	VAX	Under investigation for Alpha
Snapshot facility	VAX	Not planned for Alpha
SYSMAN I/O function for loading device drivers[1]	Alpha	Under investigation for VAX

[1]For an example of the SYSMAN I/O function for loading device drivers, see Table 1–3.

1.4.2.1 DECevent Event Management Utility

OpenVMS AXP Version 6.1 includes the DECevent utility (DECevent), which provides the interface between a system user and the system's event log files. DECevent allows system users to produce ASCII reports derived from system event entries. DECevent uses the system event log file, SYS$ERRORLOG:ERRLOG.SYS, as the default input file for event reporting unless another file is specified.

1.4.2.2 MSCP Dynamic Load Balancing

MSCP dynamic load balancing is used by VMScluster systems to balance the I/O load efficiently among systems within a VMScluster. Dynamic load balancing automatically checks server activity every 5 seconds. If activity to any server is excessive, the serving load automatically shifts to other servers in the cluster.

1.4.2.3 Installation of the Operating System with PCSI

The POLYCENTER Software Installation (PCSI) utility is provided with OpenVMS VAX and OpenVMS Alpha. On VAX, the use of PCSI is limited to installing layered products. On Alpha, PCSI is also used to install OpenVMS.

1.4.2.4 Optional Software Products Not Supported

Some optional Digital software products supported on OpenVMS VAX are not yet supported on OpenVMS Alpha. If you copy existing startup procedures from one of your OpenVMS VAX computers to an OpenVMS Alpha computer, you must comment out the calls to the startup procedures of currently unsupported optional software products.

The *Alpha Applications Catalog* lists all the available Digital optional software products and third-party applications. You can obtain a copy in the United States and Canada by calling 800-DIGITAL (800-344-4825), selecting the option for "...prepurchased product information," and speaking with a Digital representative.

In other locations, you can obtain the catalog from your Digital account representative or authorized reseller.

1.4.2.5 Patch Utility

The Patch utility (PATCH) is not offered on OpenVMS Alpha. However, investigation is underway regarding support for the PATCH/ABSOLUTE function on OpenVMS Alpha.

1.4.2.6 Snapshot Facility

The Snapshot facility (Snapshot), sometimes referred to as Fastboot, lets you reduce system startup time by booting OpenVMS VAX from a saved system image disk file. Snapshot can be used only for a standalone system (that is, a system that is not in a VMScluster environment).

For more information, see the *OpenVMS System Manager's Manual*.

1.5 Programming Environment

The similarities and differences in the programmer's environment between OpenVMS VAX and OpenVMS Alpha are outlined in this section and described in more detail in *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*, except where noted otherwise. *Migrating an Application from OpenVMS VAX to OpenVMS Alpha* also provides guidelines for developing applications that run on both OpenVMS VAX and OpenVMS Alpha and additional guidelines for developing applications that run in a mixed-architecture VMScluster.

The same types of program development tools that you are accustomed to using on OpenVMS VAX are available on OpenVMS Alpha systems including the Linker utility, the Librarian utility, the OpenVMS Debugger (also known as the symbolic debugger), the Delta/XDelta Debugger, and run-time libraries, including the parallel processing run-time library (PPL RTL).

1.5.1 RISC Architecture of Alpha

The RISC architecture of Alpha computers differs from the complex instruction set computer (CISC) architecture of VAX computers. The differences are particularly apparent when multiple processes or multiple execution threads in the same process access the same region of memory. An asynchronous system trap (AST) routine whose execution preempts the processing of a main routine is one example of concurrent threads within a single process.

The VAX architecture, through its microcode, provides instruction semantic guarantees that the Alpha architecture does not. Complex atomic operations and synchronization guarantees incur overhead that RISC architectures are designed to avoid.

For example, on a VAX computer, you can perform complex memory operations atomically and can access data at byte and word memory locations. If your code contains such VAX architectural dependencies, you likely will need to either use a compiler qualifier that mitigates the dependencies or make changes to your code.

1.5.2 User-Written Device Drivers

Both OpenVMS VAX and OpenVMS Alpha support user-written device drivers. The overall structure of device drivers is the same on each architecture but source changes are required to migrate a device driver from one architecture to the other. In addition, on OpenVMS Alpha, device drivers can be written in C.

For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.3 Compilers

Most DEC compilers are available on OpenVMS Alpha as shown in the following list:

- DEC Ada
- DEC BASIC
- DEC C
- DEC C++
- DEC COBOL
- DEC Fortran
- DEC Pascal
- MACRO–32
- DEC OPS5
- DEC PL/I

A BLISS compiler is also available for OpenVMS Alpha, although it is unsupported. It is included on the OpenVMS Freeware CD that ships with OpenVMS VAX and with OpenVMS Alpha.

These compilers are also available on OpenVMS VAX, except for the MACRO–32 compiler, which converts VAX MACRO code into Alpha machine code. The MACRO–32 compiler is bundled with OpenVMS Alpha to ease migration.

The differences between the compilers on VAX and Alpha may require some minor changes to your code. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.4 Native Assembler

The native assembler, MACRO–64, is not bundled with the OpenVMS Alpha operating system. Instead, it is available as an optional software product. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.5 DECmigrate for OpenVMS AXP

DECmigrate for OpenVMS AXP is an optional Digital software product that translates OpenVMS VAX images into OpenVMS Alpha images. If the native compiler or source code for your application is not available, you may be able to translate it, although there are a number of restrictions. See the appendix Translation and Performance Restrictions in *DECmigrate for OpenVMS AXP Systems Translating Images*.

DECmigrate for OpenVMS AXP can also be used to analyze code to determine how easy or difficult it would be to migrate it. For more information about DECmigrate for OpenVMS AXP, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.6 Linker

The way certain linking tasks, such as creating shareable images, are performed is different on OpenVMS Alpha systems. You may need to modify the LINK command used to build your application. For example, instead of creating a transfer vector file for a shareable image, you must create a linker options file and declare universal symbols by specifying the SYMBOL_VECTOR= option. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.7 Librarian

The Librarian utility provides a new qualifier, /VAX. The /VAX qualifier directs the Librarian utility to create an OpenVMS VAX object module library when used with the /CREATE and /OBJECT qualifiers or an OpenVMS VAX shareable image library when used with the /SHARE qualifier. OpenVMS Alpha libraries are the default on OpenVMS Alpha systems. For more information, see the *OpenVMS Command Definition, Librarian, and Message Utilities Manual*.

1.5.8 Debuggers

The OpenVMS Debugger provides several features that facilitate debugging OpenVMS Alpha code. These features address the architectural differences that exist between VAX and Alpha computers. For example, the /UNALIGNED_DATA qualifier used with the SET command enables you to detect unaligned data.

The OpenVMS Debugger offers a new facility, the Heap Analyzer, that represents graphically, in real time, the utilization of dynamic memory (heap). This can be used for user-mode utility or application code. For more information about the OpenVMS Debugger, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

The Delta/XDelta Debugger provides several new commands and changes to existing commands for debugging OpenVMS Alpha programs. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

The OpenVMS Alpha System-Code Debugger lets you use the familiar OpenVMS Debugger interface to observe and manipulate system code interactively as it executes. This debugger is not available on OpenVMS VAX. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.9 System Dump Analyzer

The System Dump Analyzer on OpenVMS Alpha systems is almost identical to the utility provided on OpenVMS VAX systems. Most commands, qualifiers, and displays are the same. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

The Crash Log Utility Extractor (CLUE) is available on both OpenVMS VAX and OpenVMS Alpha. It was created to make the information for debugging crash dumps more accessible and more useful. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.10 Programming Components Not Available on Both Systems

Some programming components are not available on both OpenVMS VAX and OpenVMS Alpha. They are shown in Table 1–6.

Table 1–6 Programming Components Not Available on Both Systems

Component	On VAX or Alpha	Status
H_float and D_float floating-point data types	VAX	Not planned for Alpha (see Section 1.5.10.1)
OpenVMS Alpha System-Code Debugger	Alpha	Not planned for VAX (see Section 1.5.8)
Support for device drivers written in C	Alpha	Not planned for VAX (see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*)

Before moving any OpenVMS VAX applications to an OpenVMS Alpha system, Digital recommends that you become familiar with the differences in the OpenVMS Alpha program development environment as described in this section and in *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

1.5.10.1 Floating-Point Data Types

Support for the H_float floating-point and full-precision D_float floating-point data types has been eliminated from the hardware to improve overall system performance.

Alpha hardware converts D_float floating-point data to G_float floating-point data for processing. On VAX computers, the D_float floating-point data type has 56 fraction bits (D56) and 16 decimal digits of precision.

The H_float floating-point and D_float floating-point data types can usually be replaced by the G_float floating-point data type or one of the IEEE formats. However, if you require the H_float floating-point data type or the extra precision of the D_float floating-point data type, you may have to translate part of your application with DECmigrate for OpenVMS AXP.

1.5.10.2 Vector Processing

Vector processors were an option for VAX 6500 and VAX 9000 computers to provide higher performance for numerically intensive applications. Alpha computers and later versions of VAX computers do not provide this option because their basic designs provide high-speed calculations.

If you used this option for Fortran applications, you do not have to make any changes to your code for it to run on Alpha computers or later models of VAX computers. You only have to recompile it with the DEC Fortran for OpenVMS Alpha compiler. If you used vector-specific support in VAX MACRO applications, you will need to make changes to your code before recompiling it with the MACRO–32 compiler for OpenVMS Alpha.

Image files whose source files included vector instructions cannot be translated. The VAX Environment Software Translator (VEST) component of DECmigrate does not support them.

1.6 Migration Support from Digital

Digital offers a variety of services to help you migrate your applications to OpenVMS Alpha.

Digital customizes the level of service to meet your needs. The VAX to Alpha migration services available include the following:

- Migration Assessment
- Application Migration Detailed Analysis and Design
- System Migration Detailed Analysis and Design
- Application Migration
- System Migration

To determine which services are appropriate for you, contact your Digital account representative or authorized reseller, or the Digital Systems Integration Business Development Manager for your region (AP, Americas, Europe). Call 800-832-6277 (within the United States) or 603-884-8990 (outside the United States).

1.6.1 Migration Assessment Service

The Migration Assessment service assesses the VAX system and application environment to be migrated to the Alpha platform. The objectives of the migration are reviewed and a complete current state configuration is completed. The desired end state is determined and risks and constraints are identified. Finally, several migration scenarios are developed.

1.6.2 Application Migration Detailed Analysis and Design Service

The Application Migration Detailed Analysis and Design service does a detailed analysis of an in-house developed application, creating a report of all VAX dependencies within all modules and recommendations as to what modifications should be made to migrate the application to Alpha. Acceptance criteria are specified for performance and functionality.

1.6.3 System Migration Detailed Analysis and Design Service

The System Migration Detailed Analysis and Design service performs a detailed analysis of the current system environment which includes hardware, software (Digital and third party, excluding in-house developed applications) and network components. The best tools and migration methods are determined and a project plan, which maps the steps from the current to the future state, is created.

1.6.4 Application Migration Service

The Application Migration service migrates an in-house developed application from an OpenVMS VAX platform to an Alpha platform. Each code module is either recompiled or translated depending on source code availability. VAX dependencies are removed beforehand. Finally the entire application is relinked and tested on the Alpha platform. The application is then deployed on the target system(s).

1.6.5 System Migration Service

The System Migration service migrates an OpenVMS system (single node or cluster) from the VAX platform to the Alpha platform. The customer's system availability and performance requirements are reviewed and acceptance testing methodology and criteria are determined.

1.7 Migration Training

Digital Customer Training offers several seminars and courses to provide migration training to third-party application developers and end users. The first course in the following list is designed for technical or MIS managers, and the others are designed for experienced OpenVMS VAX programmers:

- *Alpha Planning Seminar*—2 days
- *Migrating HLL Applications to OpenVMS Alpha*—3 days
- *Migrating MACRO–32 Applications to OpenVMS Alpha*—2 days

To obtain a schedule and enrollment information in the United States, call 800-332-5656. In other locations, contact your Digital account representative or authorized reseller.

2 User Issues

In your role of supporting new users on OpenVMS Alpha systems, you might encounter questions about the following additional topics:

- The Help Message utility (see Section 2.1)
- The online Bookreader documentation provided on the OpenVMS Alpha compact disc (see Section 2.2)
- DCL commands (see Sections 2.3 and 2.4)
- Differences in password generation display (see Section 2.5)
- TPU default editor for the EDIT command (see Section 2.6)
- The TECO editor (see Section 2.7)
- Shareable images in the DEC C RTL (see Section 2.8)
- Run-time libraries listed not included in this version of OpenVMS Alpha (see Section 2.9)
- Compatibility between the OpenVMS VAX and OpenVMS Alpha Mathematics Libraries (see Section 2.10)
- Linker utility enhancements (see Section 2.11)

2.1 Help Message Utility

Help Message is a versatile 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 Alpha 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.

2.2 Online Documentation on Compact Disc

The OpenVMS Extended Documentation Set is included on the OpenVMS Alpha 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 CD–ROM User's Guide* for a list of the manuals on the CD and information about enabling access to and reading the online documents.

2.3 DIGITAL Command Language (DCL)

The DIGITAL Command Language (DCL), the standard user interface to OpenVMS, remains essentially unchanged with OpenVMS Alpha. All commands and qualifiers available on OpenVMS VAX are also available on OpenVMS Alpha, except for a few, as shown in Table 2–1.

Because of architectural differences between VAX and Alpha computers, some differences exist in the implementation of DCL commands, qualifiers, and lexical functions. These differences are also noted in Table 2–1.

Table 2–1 DCL Differences Between OpenVMS VAX and OpenVMS Alpha

Command/Qualifier	On VAX	On Alpha
ANALYZE/IMAGE	System versions displayed are the versions of the system symbol table, the image that is linked against the executive.	System versions displayed are the versions of the system shareable image, the image that is linked against the executive.
	Cannot analyze an OpenVMS Alpha image.	Can analyze both OpenVMS Alpha and OpenVMS VAX images.
ANALYZE/PROCESS	You use the OpenVMS Debugger to analyze the dumped image.	In some cases, you cannot use the OpenVMS Debugger, such as when the dumped image's program counter (PC) is set to an invalid address. Instead, you can use the Delta Debugger.
CLUE	Invokes CLUE.	CLUE commands are accessed through SDA. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.
DIAGNOSE	Not available; under investigation.	Invokes the DECevent utility and selectively reports the contents of one or more event log files.
FONT	Converts an ASCII bitmap distribution format (BDF) into binary server natural form (SNF).	Converts an ASCII bitmap distribution format (BDF) into binary portable compiled format (PCF).
INITIALIZE /STRUCTURE=level	Supports Files–11 On-Disk Structure Level 1 Disks.	Does not support Files–11 On-Disk Structure Level 1 Disks.
MACRO/ALPHA	Not available.	Invokes the native MACRO–64 assembler, if installed. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

(continued on next page)

Table 2–1 (Cont.) DCL Differences Between OpenVMS VAX and OpenVMS Alpha

Command/Qualifier	On VAX	On Alpha
MACRO/MIGRATION	Not available.	Invokes the MACRO–32 compiler. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.
PATCH	Invokes the Patch utility.	Not available; limited functionality under investigation.
SET HOST/DUP	Commands for installing FYDRIVER differ from those used on Alpha.	Commands for installing FYDRIVER differ from those used on VAX.
SET PASSWORD	The random password generator differs from that on Alpha. One difference is that the passwords presented on VAX are hyphenated.	The random password generator presents passwords that are not hyphenated. For more information, see Section 1.3.9.
SET TERMINAL /PROTOCOL=DDCMP	Controls whether the terminal port specified is changed into an asynchronous DDCMP line.	Not available.
SET TERMINAL /SWITCH=DECNET	Causes the terminal lines at each node to be switched to dynamic asynchronous DDCMP lines when specified with the /PROTOCOL=DDCMP qualifier.	Not available.
SHOW MEMORY/GH_ REGIONS	Not available.	Displays information about the granularity hint regions (GHR) that have been established.
SHOW MEMORY	Each dynamic memory area displayed in bytes and pages.	Each dynamic memory area displayed in pagelets.
Working set qualifiers such as /WSDEFAULT, /WSEXTENT, and /WSQUOTA	Specified in units of 512-byte pages	Specified in units of 512-byte pagelets, rounded to the nearest CPU-specific page. For more information, see Section 3.2.1.
Lexical Functions	**On VAX**	**On Alpha**
F$CONTEXT	Base priority valid range 0–31.	Base priority valid range 0–63.
F$GETSYI	For CPU, the integer that identifies the processor type is stored in the processor's system identification (SID) register.	For CPU, the integer that identifies the processor type is stored in the hardware restart parameter block (HWRPB).
	Not available.	For CONSOLE_VERSION, returns the console firmware version of your system.
	For HW_MODEL, the integer that identifies the model type is less than or equal to 1023.	For HW_MODEL, the integer that identifies the model type is greater than 1023.
	Not available.	For PALCODE_VERSION, returns the PALcode (privileged architecture library) version of your system.

2.4 Unsupported DCL Commands

The following DCL commands are not supported on OpenVMS Alpha:

- MONITOR POOL
- SET FILE/UNLOCK

2.5 Password Generation

On OpenVMS Alpha 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 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 Alpha and is intended to accommodate possible future support for alternate-language password generation databases.

2.6 Default Editor for EDIT Command

The default editor for the EDIT command on OpenVMS Alpha (and on OpenVMS VAX V6.*n*) is TPU. The default editor on VAX VMS Version 5.*n* and earlier releases 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 Alpha.

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 Alpha logical name EVE$KEYPAD to be EDT. See the *OpenVMS AXP Version 6.1 Release Notes* for details about how to do this at either the process level or the system level. You can find an example of how to define EVE$KEYPAD at the system level in the file SYS$STARTUP:SYLOGICALS.TEMPLATE.

The general release notes chapter of the *OpenVMS AXP Version 6.1 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.

2.7 TECO Editor

The TECO editor is included in OpenVMS Alpha. 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*.

2.8 Shareable Images in the DEC C RTL for OpenVMS Alpha

If you answer problem reports submitted by programmers who are coding C applications on OpenVMS Alpha systems, you might receive questions about the shareable images in the DEC C Run-Time Library (RTL).

On Alpha 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.

On an Alpha system that has DEC C installed, type "HELP CC /PREFIX" at the DCL prompt for a description of the prefixing behavior of the compiler. To resolve nonprefixed names, programmers can link against object libraries SYS$LIBRARY:VAXCRTL.OLB, SYS$LIBRARY:VAXCRTLD.OLB, SYS$LIBRARY:VAXCRTLT.OLB, and SYS$LIBRARY:VAXCCURSE.OLB. On VAX systems, those libraries contain object code for the RTL support; on Alpha systems, those libraries contain jacket routines to the prefixed entry points.

Note that on VAX systems, you use an option file when using VAXCRTL.EXE. On Alpha 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 language documentation for more information.

2.9 Run-Time Libraries

Table 2–2 lists the run-time libraries that are not included in OpenVMS Alpha.

Table 2–2 Run-Time Libraries Not Included in OpenVMS Alpha

DBGSSISHR	DEBUG item is not required on OpenVMS Alpha.
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 are available in OpenVMS VAX are available in OpenVMS Alpha. The OpenVMS VAX libraries that are not available are either not being ported to OpenVMS Alpha or are planned for a later release of OpenVMS Alpha.

For example, the vector math libraries VBLAS1RTL and VMTHRTL are not available in OpenVMS Alpha because there is no support on OpenVMS Alpha for programs that use the OpenVMS VAX vector instructions.

2.10 Compatibility Between the OpenVMS VAX and OpenVMS Alpha 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 Alpha system. Jacket routines are provided that map MTH$ routines to their math$ counterparts in the Digital Portable Mathematics Library (DPML) for OpenVMS Alpha. However, there is no support in the DPML for calls made to JSB entry points and vector routines. 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 *Digital Portable Mathematics Library* manual for more information about DPML.

2.11 Linker Utility Enhancements

The following sections describe new Alpha enhancements to the OpenVMS Linker utility. Refer to the online version of the *OpenVMS Linker Utility Manual* for additional information on these enhancements.

2.11.1 New /DSF Qualifier

The /DSF qualifier directs the linker to create a file called a debug symbol file (DSF) for use by the OpenVMS Alpha System-Code Debugger. The default is /NODSF. The /DSF qualifier can be used with the /NOTRACEBACK qualifier to suppress the appearance of SYS$IMGSTA in the image's transfer array. The /DSF qualifier has no effect on the contents of the image, including the image header.

The /DSF and /DEBUG qualifiers are not mutually exclusive. However, the combination is not generally useful. The debug bit in the image header will be set and SYS$IMGSTA will be included in the transfer array, but there will be no information for the symbolic debugger included in the image. The DSF file will be generated as usual.

Use the following format:

```
LINK/DSF[=file-spec]
```

See *OpenVMS Alpha Device Support: Developer's Guide* for guidelines on using the OpenVMS Alpha System-Code Debugger.

2.11.2 New /ATTRIBUTES Qualifier for COLLECT= Option

The COLLECT= option directs the linker to place the specified program section (or program sections) into the specified cluster. A new qualifier, /ATTRIBUTES, directs the linker to mark the cluster *cluster-name* with the indicated qualifier keyword value. This qualifier is used to build Alpha drivers.

Use the following format:

```
COLLECT=cluster-name [/ATTRIBUTES={RESIDENT | INITIALIZATION_CODE}],-
  psect-name[,...])
```

Qualifier Values

RESIDENT
Marks the cluster *cluster-name* as RESIDENT so that the image section created from that cluster has the EISD$V_RESIDENT flag set. This causes the loader to map the image section into nonpaged memory.

INITIALIZATION_CODE
Marks the cluster *cluster-name* as INITIALIZATION_CODE so that the image section created from that cluster has the EISD$V_INITALCOD flag set. The initialization code will be executed by the loader. This keyword is specifically intended for use with program sections from modules SYS$DOINIT and SYS$DRIVER_INIT in STARLET.OLB.

See *OpenVMS Alpha Device Support: Developer's Guide* for guidelines on using this qualifier.

3 Overview of System Management Issues

Most of the OpenVMS VAX system management utilities, command formats, and tasks are identical in the OpenVMS Alpha environment. There are some differences that must be considered to set up, maintain, secure, and optimize OpenVMS Alpha systems properly and to establish network connections.

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 Alpha system management.

A goal of this manual is to help you manage Alpha and VAX nodes at the same time.

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

- Chapter 4 compares the *setup* features and tasks.
- Chapter 5 compares the *maintenance* features and tasks.
- Chapter 6 compares the *security* features and tasks.
- Chapter 7 compares the *performance optimization* features and tasks.
- Chapter 8 compares the *network* management features and tasks.

Chapter 2 describes additional considerations related to your task of supporting general users and programmers on OpenVMS Alpha systems.

3.1 Reduced Number of System Management Differences

Starting with OpenVMS AXP Version 6.1, a number of key features and related products were present, including:

- Fully functional VMScluster systems. The same features in VAXclusters are available in VMSclusters. VMSclusters offer the additional capability of supporting dual architectures, Alpha and VAX.
- Volume Shadowing for OpenVMS.
- RMS Journaling for OpenVMS.
- Support for multiple queue managers.
- User-written device drivers.
- The same C2 security features as in OpenVMS VAX Version 6.0 , which are certified by the U.S. government as C2 compliant. OpenVMS VAX Version 6.1 has been certified as C2 compliant at this time, and OpenVMS AXP is currently being reviewed. As pointed out in Chapter 6, the OpenVMS Alpha C2 features do not include DECnet connection auditing.

- The POLYCENTER Software Installation utility (PCSI), which provides for rapid installation or deinstallation of products and for managing information about the products installed on your systems.
- The movefile subfunction for atomic-file disk-defragmentation applications, and support of the layered software product DEC File Optimizer for OpenVMS Alpha.

3.2 Why Some Differences Still Exist

Why do some system management differences continue to exist in OpenVMS Alpha and OpenVMS VAX environments? The following sections summarize the significant reasons for these differences. The remaining chapters in this document provide more details and will help you identify the OpenVMS Alpha and OpenVMS VAX system management characteristics.

3.2.1 Different Page Size

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

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

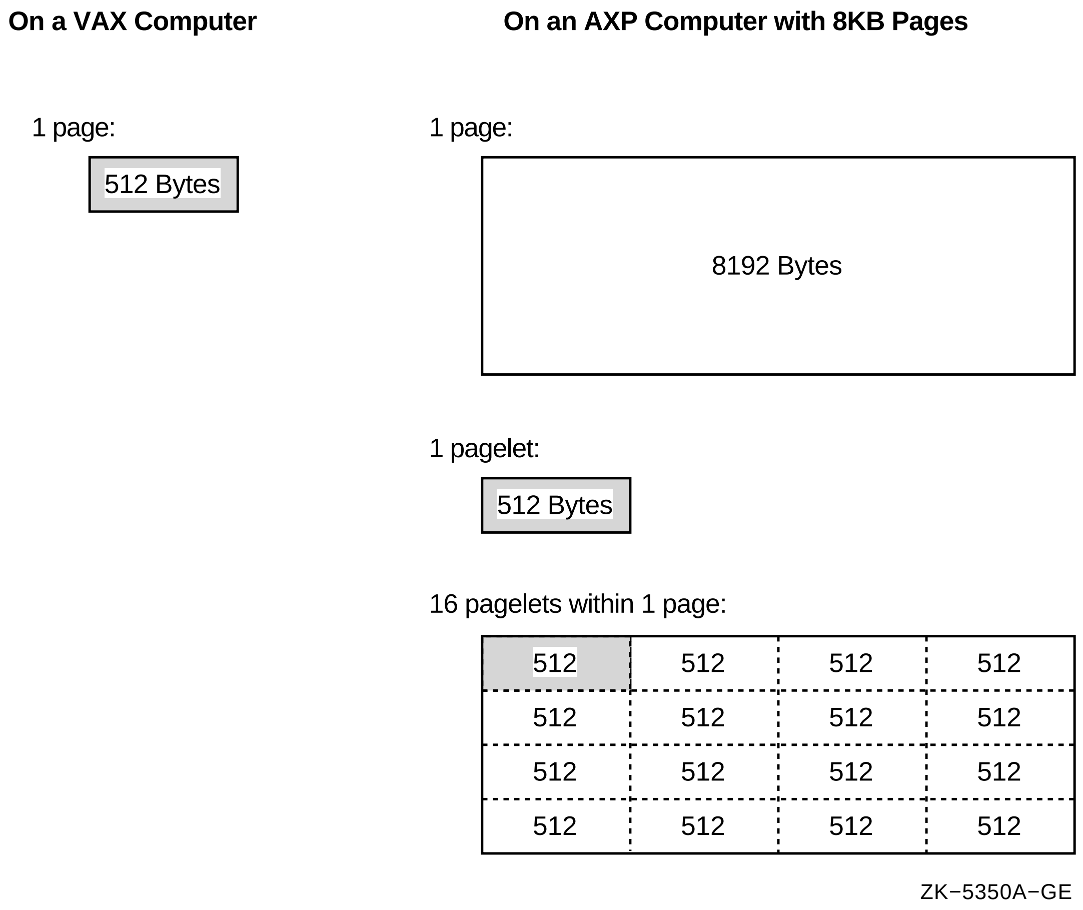
- You might need to adjust process quotas and limits, as well as system parameters, to account for the additional resources (especially memory resources) users might require. For example, higher values may be necessary for the PGFLQUOTA process quota and the GBLPAGES system parameter.
- In a number of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

  Internally, for the purposes of memory allocation, deletion, and protection, OpenVMS Alpha rounds 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 VMScluster, which can include certain versions of OpenVMS VAX nodes and OpenVMS Alpha nodes, it is helpful to know that a VAX page and an Alpha 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 Alpha.

Figure 3–1 illustrates the relative sizes of a VAX page, an Alpha 8KB page, and an Alpha pagelet.

Figure 3–1 VAX Page Size, Alpha Page Size, and Alpha Pagelet Size



OpenVMS Alpha 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 Alpha pagelet value equal a common 512-byte quantity.

In your OpenVMS Alpha environment, you need to notice when page or pagelet values are being shown in memory displays. If a memory value represents a page on an Alpha system, 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 Alpha computer in use (8KB pages, 16KB pages, 32KB pages, or 64KB pages). In general, OpenVMS Alpha 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 4.2.21, Section 7.1, Section 7.2, and Section 7.3.

3.2.2 A New Way to Perform Standalone Backups and Other Tasks

On Alpha systems, you can use a menu-driven procedure (which starts when you boot the OpenVMS Alpha operating system distribution compact disc) to perform the following tasks:

- Enter a DCL environment, from which you can perform backup and restore operations on the system disk.
- Install or upgrade the operating system, using the POLYCENTER Software Installation utility.

For more detailed information about using the menu-driven procedure, see the *OpenVMS Alpha Upgrade and Installation Manual* and the *OpenVMS System Manager's Manual*.

3.2.3 DECevent Event Management Utility

OpenVMS Alpha includes the DECevent utility, which provides the interface between a system user and the system's event log files. This allows system users to produce ASCII reports derived from system event entries. DECevent uses the system event log file, SYS$ERRORLOG:ERRLOG.SYS, as the default input file for event reporting unless another file is specified.

Note

The DECevent utility is not available on OpenVMS VAX systems.

See Section 5.2.1 for a summary of the DECevent utility.

3.2.4 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 Alpha computers, you use the System Management utility (SYSMAN) instead of SYSGEN. OpenVMS Alpha 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 Alpha systems as part of your migration effort. See Chapter 4 and the *OpenVMS System Management Utilities Reference Manual: M–Z* for details about SYSMAN IO commands.

3.2.5 MONITOR POOL Command Not Necessary

The DCL command MONITOR POOL that is used on VMS Version 5.5 and earlier releases is not provided on OpenVMS Alpha systems or on OpenVMS VAX Version 6.0 and later releases. MONITOR POOL functions are no longer needed due to adaptive pool management features present in the operating system. See Section 7.4 for details about adaptive pool management.

[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).

3.2.6 Changes in OpenVMS File Names

The names of some command procedure files supplied by the operating system changed. For example, SYSTARTUP_V5.COM from VMS Version 5.5 and earlier releases is called SYSTARTUP_VMS.COM on OpenVMS Alpha and on OpenVMS VAX Version 6.0 and later releases. Also, the VAXVMSSYS.PAR system parameter file is called ALPHAVMSSYS.PAR on OpenVMS Alpha. See Chapter 4.

3.2.7 Unsupported Features in DECnet for OpenVMS Alpha

Some networking features in DECnet for OpenVMS VAX (Phase IV software) are not available with DECnet for OpenVMS Alpha. These unsupported features on OpenVMS Alpha systems are:

- The DECnet for OpenVMS DECdns node name interface.
- VAX P.S.I.; however, a DECnet for OpenVMS Alpha node can communicate with DECnet nodes that are connected to X.25 networks reachable via a DECnet/X.25 router.
- Level 2 host-based DECnet routing. Level 1 routing is supported for use only with cluster alias routers and is restricted for use on one circuit.
- DDCMP network connections.

Note

If you install the version of DECnet/OSI that is for OpenVMS AXP Version 6.1:

- DECdns client is available.
- X.25 support is available through DEC X.25 for OpenVMS Alpha. The QIO interface drivers from the P.S.I. product are included in DEC X.25 in order to provide the same interface to customer applications.

4 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 Alpha and VAX computers
- Explains which OpenVMS system management setup tasks are different or new on Alpha computers

4.1 Setup Tasks That Are the Same

Table 4–1 lists the OpenVMS system management setup tasks that are identical or similar on Alpha and VAX computers.

Table 4–1 Identical or Similar OpenVMS Setup Tasks

Feature or Task	Comments
System disk directory structure	Directory structure is the same (SYS$SYSTEM, SYS$MANAGER, SYS$UPDATE, and so on).
Site-independent STARTUP.COM command procedure	Each OpenVMS release ships a new SYS$SYSTEM:STARTUP.COM procedure. Do not modify STARTUP.COM.
Site-specific startup command procedures	OpenVMS 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 Alpha and in OpenVMS VAX Version 6.0 and later releases.
VMSINSTAL procedure	Although the VMSINSTAL procedure is still available on OpenVMS Alpha systems, updated layered products will use the POLYCENTER Software Installation utility (PCSI) instead. PCSI is available for rapid installations and deinstallations, and for managing information about installed products on OpenVMS Alpha and OpenVMS VAX systems. See Section 4.2.2 for a comparison of VMSINSTAL and PCSI.
Decompressing libraries as a postinstallation task	Use @SYS$UPDATE:LIBDECOMP.COM as usual if you choose to decompress the system libraries (recommended).

(continued on next page)

Table 4–1 (Cont.) Identical or Similar OpenVMS Setup Tasks

Feature or Task	Comments
VAXcluster system	VMScluster systems on OpenVMS Alpha include all the features available in VAXcluster systems. As with VAXcluster systems, there are limits to: the number of nodes in a VMScluster, the operating system versions that can operate together, and the layered products that can operate together. VMScluster systems offer the additional capability of supporting the Alpha and VAX architectures. Essentially, all features are common between VAXcluster systems and VMScluster systems. They include: • Shared file system—All systems share read and write access to disk files in a fully coordinated environment. (However, in a VMScluster system with VAX and Alpha nodes, each architecture needs its own system disk for booting.) • Shared batch and print queues are accessible from any system in the VMScluster system. • OpenVMS lock manager system services operate for all nodes in a VMScluster system. • All physical disks in a VMScluster system can be made accessible to all systems. (However, in a VMScluster system with both VAX and Alpha nodes, each architecture needs its own system disk for booting.) • Alpha and VAX processors can serve TMSCP tapes to all CPUs (Alpha and VAX). • Process information and control services are available to application programs and system utilities on all nodes in a VMScluster system. • Configuration command procedures assist in adding and removing systems and in modifying their configuration characteristics. • An availability manager for VMSclusters (DECamds) is optionally installable for monitoring Alpha or VAX nodes in the cluster. All features are supported for Alpha nodes, with an exception: the DECamds console application cannot run on an Alpha node. • Booting of Alpha or VAX satellite nodes can be done from an Alpha server node or a VAX server node. • The Show Cluster utility displays the status of all VMScluster hardware components and communication links. • Standard OpenVMS system and security features work in a VMScluster system such that the entire VMScluster system can operate as a single security domain. However, you can still have multiple authorization files, and so on. • The VMScluster software balances the interconnect I/O load in VMScluster configurations that include multiple interconnects. • Multiple VMScluster systems can be configured on a single or extended local area network (LAN).

(continued on next page)

Table 4–1 (Cont.) Identical or Similar OpenVMS Setup Tasks

Feature or Task	Comments
	• The /CLUSTER qualifier to the Monitor utility operates across the entire VMScluster. • VMScluster systems support the following interconnects for Alpha computers: – CI computer interconnect – DIGITAL Storage System Interconnect (DSSI) – Ethernet – FDDI A maximum of 96 systems can be configured in a VMScluster system, the same limit as in a VAXcluster system. For configuration limits, refer to the VMScluster Software for OpenVMS Alpha *Software Product Description* (SPD 42.18.*xx*). See *VMScluster Systems for OpenVMS* for detailed information about VMScluster systems.
Volume Shadowing for OpenVMS	With this release of OpenVMS Alpha, the Volume Shadowing for OpenVMS product is identical on Alpha and VAX systems. Minor exceptions are noted in *Volume Shadowing for OpenVMS*.
RMS Journaling for OpenVMS	Identical. Consequently, the following commands are available on OpenVMS Alpha nodes: • SET FILE/AI_JOURNAL • SET FILE/BI_JOURNAL • SET FILE/RU_ACTIVE • SET FILE/RU_FACILITY • SET FILE/RU_JOURNAL • RECOVER/RMS_FILE
User-written device drivers	Similar on VAX and Alpha. OpenVMS AXP Version 1.5 included a mechanism to allow essential OpenVMS VAX drivers to be ported to Alpha systems quickly and with minimal changes. This mechanism, known as the Step 1 driver interface, provided device support for customers with critical needs. The Step 1 driver interface was replaced by the Step 2 driver interface in OpenVMS AXP Version 6.1. The Step 2 interface design facilitates the coding and ensures a longer life for any new device drivers. Unlike Step 1 drivers, Step 2 drivers can be written in C and can conform to the OpenVMS calling standard. Any Step 1 device driver for earlier versions of OpenVMS Alpha must be replaced with Step 2 drivers on OpenVMS AXP Version 6.1 or later. For more information about Step 2 drivers, see *Creating an OpenVMS AXP Step 2 Device Driver from a Step 1 Device Driver*.

(continued on next page)

Table 4–1 (Cont.) Identical or Similar OpenVMS Setup Tasks

Feature or Task	Comments
Backing up data	The BACKUP command and qualifiers are the same on OpenVMS Alpha. Important: Thoroughly read the *OpenVMS AXP Version 6.1 Release Notes* for the latest information about any restrictions with backup and restore operations on Alpha and VAX systems.
LAT startup	You must start DECnet before you start LAT. As on OpenVMS VAX, always start LAT from the SYSTEM account. This account 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. Set the BYTLM quota for accounts that use LATCP 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. You can enable service responder and outgoing connections on an Alpha computer.
STARTUP SET OPTIONS and SHUTDOWN NODE commands in SYSMAN	Identical.
Digital's InfoServer, which includes the Local Area Disk Control Program (LADCP), LASTport network transport control program, LASTport /Disk protocol, LASTport /Tape protocol	Identical.
Security audit log file name	The same name on OpenVMS Alpha and OpenVMS VAX: SECURITY.AUDIT$JOURNAL. Called SECURITY_AUDIT.AUDIT$JOURNAL on earlier OpenVMS Alpha releases and on VAX VMS Version 5.5 and earlier releases. Resides in SYS$COMMON:[SYSMGR] in both cases.
DECdtm and its two-phase commit protocol	Identical.
VMSTAILOR and DECW$TAILOR utilities	Identical.
Authorize utility	Identical.
OPCOM	Identical.
Terminal Fallback Facility (TFF)	Similar function but with some differences. See Section 4.2.22 for details.

(continued on next page)

Table 4–1 (Cont.) Identical or Similar OpenVMS Setup Tasks

Feature or Task	Comments
User Environment Test Package (UETP)	Identical.

4.2 Setup Tasks That Are Different

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

- On OpenVMS Alpha systems, there is a new way to back up a system disk (as an alternative to Standalone BACKUP, which is used on OpenVMS VAX systems) (see Section 4.2.1).
- Installations are different, depending on whether the product you are installing uses the VMSINSTAL procedure or the new POLYCENTER Software Installation utility (see Section 4.2.2).
- You must plan for and manage common object and image file extensions, especially in mixed-architecture VMSclusters (see Section 4.2.3).
- The format of the BOOT command on Alpha systems and the boot flags are different (see Section 4.2.4).
- The CONSCOPY.COM command procedure is not supplied with the OpenVMS Alpha kit (see Section 4.2.5).
- Use of the License Management Facility (LMF) is different (see Section 4.2.6).
- One of the two Product Authorization Keys (PAKs) for DECnet for OpenVMS Alpha has a different name from one of the PAK names on VAX systems (see Section 4.2.7).
- The PAK name for VMSclusters is different from the PAK name for VAXclusters (see Section 4.2.8).
- System Generation utility (SYSGEN) and its parameters are different (see Section 4.2.9).
- I/O subsystem configuration commands, controlled in SYSGEN on OpenVMS VAX, are in the OpenVMS Alpha System Management utility (SYSMAN) (see Section 4.2.10).
- Symmetric multiprocessing (SMP) can be slightly different (see Section 4.2.11).
- Startup command procedure changes because of relocation of I/O subsystem configuration functions from SYSGEN to SYSMAN (see Section 4.2.12).
- Hardware devices supported by Alpha computers may vary from those supported by VAX computers (see Section 4.2.13).
- Local DIGITAL Storage Architecture (DSA) device naming is different (see Section 4.2.14).
- The file name format of drivers supplied by Digital on Alpha systems is different (see Section 4.2.15).
- OpenVMS Alpha installation media (CD–ROM) contains binaries and documentation (see Section 4.2.16).

- VMSINSTAL utility on OpenVMS Alpha systems includes features not available on VAX systems (see Section 4.2.17).
- Running the AUTOGEN procedure is different on OpenVMS Alpha systems (see Section 4.2.18).
- You can improve the performance of main images and shareable images on OpenVMS Alpha systems by using a feature called granularity hint regions (see Section 4.2.19).
- On OpenVMS Alpha systems, the SYS.EXE loadable executive image has been renamed to SYS$BASE_IMAGE.EXE (see Section 4.2.20).
- On OpenVMS Alpha systems, a rounding-up algorithm is used when you input values for quotas that are in quantities of pagelets (see Section 4.2.21).
- The Terminal Fallback Facility (TFF) is different (see Section 4.2.22).

4.2.1 New Way to Back Up System Disks

On Alpha systems, you can use a menu-driven procedure (which starts when you boot the OpenVMS Alpha operating system distribution compact disc) to perform the following tasks:

- Enter a DCL environment, from which you can perform backup and restore operations on the system disk.
- Install or upgrade the operating system, using the POLYCENTER Software Installation utility.

For more detailed information about using the menu-driven procedure, see the *OpenVMS AXP Version 6.1 Upgrade and Installation Manual*. See Section 4.2.2 for a summary of the POLYCENTER Software Installation utility.

4.2.2 VMSINSTAL and the POLYCENTER Software Installation Utility

Although the VMSINSTAL procedure is still available on OpenVMS Alpha systems, updated layered products will use the POLYCENTER Software Installation utility instead. The POLYCENTER Software Installation utility is available for rapid installations and deinstallations, and for managing information about installed products on both OpenVMS Alpha and OpenVMS VAX systems. OpenVMS AXP Version 1.5 and earlier systems, and OpenVMS VAX Version 6.0 and earlier systems all use VMSINSTAL for installations.

Table 4–2 compares the VMSINSTAL procedure and the POLYCENTER Software Installation utility.

Table 4–2 Comparison of VMSINSTAL and POLYCENTER Software Installation Utility

VMSINSTAL Procedure	POLYCENTER Software Installation Utility
Begins an installation before you make configuration choices.	You can perform an installation by choosing your software configuration and then installing the product.

(continued on next page)

Table 4–2 (Cont.) Comparison of VMSINSTAL and POLYCENTER Software Installation Utility

VMSINSTAL Procedure	POLYCENTER Software Installation Utility
When you press Ctrl/Y, the procedure leaves your system in the same state it was in prior to the start of the installation.	If you press Ctrl/Y (or click on the Cancel button) while the POLYCENTER Software Installation utility is installing a product, you must perform a remove operation to clean up any files created by the partial installation.
Prompts you as to whether you want to back up your system disk.	Does not ask you whether you want to back up your system disk before you begin an installation.

See the following documents for detailed information about the POLYCENTER Software Installation utility:

- *POLYCENTER Software Installation Utility User's Guide*
- *POLYCENTER Software Installation Utility Developer's Guide*

4.2.3 Planning for and Managing Common Object and Image File Extensions

File extensions on OpenVMS VAX computers are identical on OpenVMS Alpha 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 VMSclusters that include Alpha and VAX systems using common disks:

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

Determining the Host Architecture

Use the F$GETSYI lexical function from within your command procedure, or the $GETSYI system service or the LIB$GETSYI RTL routine from within your program, to determine whether the procedure or program is running on an OpenVMS VAX system or on an OpenVMS Alpha system.

Example 4–1 illustrates how to determine the host architecture in a DCL command procedure by calling the F$GETSYI lexical function and specifying the ARCH_TYPE item code. For an example of calling the $GETSYI system service in an application to determine the page size of an Alpha computer, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

Example 4–1 Using ARCH_TYPE to Determine Architecture Type

```
$! Determine Architecture Type
$!
$ Type_symbol = f$getsyi("ARCH_TYPE")
$ If Type_symbol .eq. 1 then goto ON_VAX
$ If Type_symbol .eq. 2 then goto ON_Alpha
$ ON_VAX:
$ !
$ ! Do VAX-specific processing
$ !
$ Exit
$ ON_Alpha:
$ !
$ ! Do Alpha-specific processing
$ !
$ Exit
```

Once the architecture type is determined, the command procedure or program can branch to the appropriate code for architecture-specific processing.

The ARCH_TYPE argument and other useful F$GETSYI arguments are summarized in Table 4–3.

Table 4–3 F$GETSYI Arguments That Specify Host Architecture

Argument	Usage
ARCH_TYPE	Returns “1” on a VAX computer; returns “2” on an Alpha computer
ARCH_NAME	Returns “VAX” on a VAX computer; returns “Alpha” on an Alpha computer
PAGE_SIZE	Returns “512” on a VAX computer; returns “8192” on an Alpha computer

Example 4–2 shows a simple command procedure that uses F$GETSYI to display several architecture-specific values.

Example 4–2 Using F$GETSYI to Display Hardware Type, Architecture Type, and Page Size

```
$ ! 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 an OpenVMS VAX node, output from the procedure is similar to the following display:

```
OpenVMS process with PID 3FE00B0E
 running at 16-MAY-1994 04:23:07.92.

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 Alpha node, output from the procedure is similar to the following display:

```
OpenVMS process with PID 2FC00126
 running at 16-MAY-1994 04:23:59.37.

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$GETSYI lexical function, the PAGE_SIZE, ARCH_NAME, and ARCH_TYPE arguments do not exist on VMS systems predating Version 5.5. On VMS Version 5.4 and earlier systems, you can use the F$GETSYI("CPU") lexical function.

Output of Analyze Utility Commands

The display created by ANALYZE/OBJECT and ANALYZE/IMAGE commands on an OpenVMS Alpha node identifies the architecture type of an .OBJ or .EXE file. The OpenVMS Alpha commands ANALYZE/OBJECT and ANALYZE/IMAGE work with Alpha or VAX object 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 Alpha 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 Alpha 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 Alpha node and the image being analyzed is an OpenVMS Alpha 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 Alpha node and the object being analyzed is an OpenVMS Alpha 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 Alpha compiler or linker. For example:

```
$ ! On an OpenVMS VAX node
$ RUN SALARY_REPORT.EXE     !  An OpenVMS Alpha 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 Alpha node. For example:

```
$ ! On an OpenVMS Alpha 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
```

4.2.4 BOOT Console Command

The Alpha 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 Alpha 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*.

The *boot_flags* parameter specifies the hexadecimal representation of the sum of the desired boot flags. Table 4–4 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 4–4 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. See Section 4.2.4.1 for more information.
20000	USER_MSGS	Enable descriptive mode, presenting a subset of the verbose mode seen when DBG_INIT is enabled. See Section 4.2.4.1 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.

4.2.4.1 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 Alpha 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 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 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
```

4.2.5 CONSCOPY.COM Command 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 Alpha computers because the Alpha consoles exist in read-only memory and not on disks.

4.2.6 Use of the License Management Facility (LMF)

Availability Product Authorization Keys (PAKs) are available for OpenVMS Alpha. You can identify an OpenVMS Alpha PAK by the keyword ALPHA in the PAK's option field.

LMF Caution for OpenVMS VAX Systems Predating Version 6.1

Note

The following cautionary text is for systems using a common license database (LDB), and involves VAX nodes that are running OpenVMS VAX Version 6.0 and earlier releases. OpenVMS VAX Version 6.1 fixes the limitations noted in the following discussion.

You can load and use PAKs having the ALPHA option only on OpenVMS Alpha systems. However, they can safely reside in a license database (LDB) shared by both OpenVMS VAX and OpenVMS Alpha systems.

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

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

By default, all OpenVMS Alpha PAKs look disabled to an OpenVMS VAX Version 6.0 or earlier system. Never use the DELETE/STATUS=DISABLED command from an OpenVMS VAX system on an LDB that contains OpenVMS Alpha PAKs. If you do, all OpenVMS Alpha 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 Alpha PAK while using an OpenVMS VAX Version 6.0 or earlier system, the PAK and the database probably will not be affected adversely. Repeat the command using LMF running on an OpenVMS Alpha system; the PAK should return to a valid state.

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

However, if you attempt to use any of the commands listed previously on a PAK located in an LDB that is shared with an OpenVMS VAX Version 6.0 or earlier system, the following serious problems may result:

- Because OpenVMS Alpha PAKs look disabled to an OpenVMS VAX Version 6.0 or earlier 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 Alpha system, the OpenVMS VAX Version 6.0 or earlier system may attempt to load the OpenVMS Alpha 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 might not load. In this case, system users may be denied access to the product.

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

As noted previously, the LMF that is part of OpenVMS VAX Version 6.1 removes these command restrictions.

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

4.2.7 PAK Name Difference Using DECnet for OpenVMS Alpha

The DECnet cluster alias feature is available on OpenVMS Alpha. Note, however, that the PAK name enabling cluster alias routing support on OpenVMS Alpha (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 Alpha 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 Alpha nodes.

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

See Chapter 8 for more information about DECnet for OpenVMS Alpha.

4.2.8 PAK Name Difference for VMSclusters and VAXclusters

The PAK name for VMSclusters is VMSCLUSTER. The PAK name for VAXclusters is VAXCLUSTER.

4.2.9 SYSGEN and System Parameters

The OpenVMS Alpha 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 Alpha SYSGEN and OpenVMS VAX SYSGEN differ in the following ways:

- OpenVMS Alpha includes several new and modified system parameters. Some of the system parameter changes are because of new features. Other changes are due to the larger page sizes of Alpha computers. See Section 4.2.9.1 through Section 4.2.9.4 for information about the new system parameters; also see Chapter 7 for information about changes to system parameters.
- On OpenVMS Alpha, I/O subsystem configuration capabilities have been removed from SYSGEN. The System Management utility (SYSMAN) provides this functionality on OpenVMS Alpha.

  Refer to Section 4.2.10 and to *OpenVMS System Management Utilities Reference Manual: M–Z* for more information about the SYSMAN I/O subsystem configuration commands.

4.2.9.1 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 Alpha and VAX computers.

On OpenVMS Alpha systems and on OpenVMS VAX Version 6.0 and later releases, the MULTIPROCESSING system parameter has a new value (4). When MULTIPROCESSING is set to 4, OpenVMS always loads the streamlined multiprocessing synchronization image, regardless of system configuration or CPU availability.

See Section 4.2.11 for more information about SMP.

4.2.9.2 PHYSICAL_MEMORY System Parameter

OpenVMS Alpha 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).

4.2.9.3 POOLCHECK System Parameter

The adaptive pool management feature described in Section 7.4 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.

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

These features are available on systems running OpenVMS Alpha 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 7.4 for more information.

4.2.9.4 New Granularity Hint Region System Parameters

Five system parameters are associated with the granularity hint regions (GHR) feature described in Section 7.5. Refer to Section 4.2.19 and Section 7.5 for more information.

4.2.10 Using SYSMAN to Configure the I/O Subsystem

Use the System Management utility (SYSMAN) on OpenVMS Alpha 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>
```

The *OpenVMS System Management Utilities Reference Manual* 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 4–5 compares the I/O subsystem configuration commands on OpenVMS Alpha and OpenVMS VAX.

Table 4–5 Comparison of I/O Subsystem Configuration Commands

OpenVMS VAX SYSGEN Command	OpenVMS Alpha 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 Alpha.
CONNECT/ADAPTER requires CMKRNL privilege only.	IO CONNECT requires CMKRNL and SYSLCK privileges.
CONNECT/ADAPTER provides the /ADPUNIT qualifier.	No equivalent.
CONNECT/ADAPTER provides the /CSR_OFFSET qualifier.	Use IO CONNECT/ADAPTER/CSR. Note: CSR is the control and status register.
CONNECT/ADAPTER provides the /DRIVERNAME (no underscore) qualifier.	IO CONNECT provides the /DRIVER_NAME qualifier.
No equivalent.	IO CONNECT provides the /LOG=(ALL,CRB,DDB,DPT,IDB,SC,UCB) qualifier and options.
CONNECT/ADAPTER provides the /MAXUNITS (no underscore) qualifier.	IO CONNECT provides the /MAX_UNITS qualifier.

[1]All I/O subsystem configuration commands on OpenVMS Alpha are preceded by “IO”.

(continued on next page)

Table 4–5 (Cont.) Comparison of I/O Subsystem Configuration Commands

OpenVMS VAX SYSGEN Command	OpenVMS Alpha SYSMAN Command[1]
No equivalent.	IO CONNECT provides the /NUM_UNITS qualifier.
CONNECT/ADAPTER provides the /NUMVEC (no underscore) qualifier.	IO CONNECT provides 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.
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 Alpha 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.	IO SHOW BUS 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.	IO SHOW DEVICE does not show start and end address information.
SHOW/DRIVER displays the start and end addresses of device drivers loaded into the system.	IO SHOW DRIVER 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 Alpha processors.

[1]All I/O subsystem configuration commands on OpenVMS Alpha 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 Alpha SYSMAN. Next, change the DCL procedures (if you copied any over from the VAX to the Alpha 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 4–5.

Note

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

4.2.11 Symmetric Multiprocessing on Alpha Systems

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

On the supported Alpha 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 hexadecimal* 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 is managed on Alpha 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, 3, and 4 have *nearly* equivalent functions on Alpha systems and on VAX systems running OpenVMS VAX Version 6.0 and later releases. Table 4–6 summarizes the functions of the five MULTIPROCESSING values.

Table 4–6 MULTIPROCESSING Values on Alpha and VAX Systems

Value	Function
0	Load the uniprocessing synchronization image.
1	Load the full-checking multiprocessing synchronization image if the CPU type is capable of SMP and two or more CPUs are present on the system.
2	Always load the full-checking version, regardless of the system configuration or CPU availability.
3	On VAX, load the SPC SYSTEM_SYNCHRONIZATION image. On Alpha, load the streamlined multiprocessing synchronization image. In either case, the load occurs only if the CPU type is capable of SMP and two or more CPUs are present on the system.
4	On VAX, always load the regular streamlined image. On an OpenVMS Alpha system, always load the streamlined multiprocessing synchronization image. In either case, the load occurs 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 System Dump Analyzer (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 Alpha 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_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 100,000 10-microsecond intervals (1 second) is usually adequate.

- 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 3,000,000 10-microsecond intervals (30 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 system to hang. A timeout causes a CPUSANITY bugcheck. For SMP_SANITY_CNT, the default value of 300 10-millisecond intervals (3 seconds) is usually adequate.

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 Alpha and VAX systems running SMP.

However, when executed on an Alpha system, the SHOW CPU/FULL command output contains information not found in the display from a VAX SMP system. In the following VAX example, the SHOW CPU/FULL command produces a configuration summary of the VAX 6000 Model 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 five-CPU Alpha system:

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

LSR4, a DEC 7000 Model 650
Multiprocessing is ENABLED. Streamlined synchronization image loaded.
Minimum multiprocessing revision levels: CPU = 1
```

```
System Page Size = 8192
System Revision Code =  A01
System Serial Number = 123456
Default CPU Capabilities:
 QUORUM RUN
Default Process Capabilities:
 QUORUM RUN

PRIMARY CPU = 00

CPU 00 is in RUN state
Current Process: *** None ***
Serial Number: 1234567
Revision:
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.44
  PALcode Compatibility = 1
  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 = 00000404  Reason: PRIMARY Capability

CPU 01 is in INIT state
Current Process: *** None ***
Serial Number: SG235LUF74
Revision: L06
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.44
  PALcode Compatibility = 1
  Maximum Shared Processors = 8
  Memory Space: Physical address = 00000000 00000000
   Length = 16
  Scratch Space: Physical address = 00000000 00020000
   Length = 16
Capabilities of this CPU:
        *** None ***
Processes which can only execute on this CPU:
        *** None ***

CPU 02 is in RUN state
Current Process: VMSADU         PID = 00000411
Serial Number: GA24847575
Revision: 06
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.44
  PALcode Compatibility = 1
  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 04 is in RUN state
Current Process: *** None ***
Serial Number: GA24847577
Revision: 06
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.44
  PALcode Compatibility = 1
  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 05 is in RUN state
Current Process: *** None ***
%SMP-F-CPUTMO, CPU #01 has failed to leave the INITIALIZATION state
Serial Number: SG237LWL46
Revision:
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.44
  PALcode Compatibility = 1
  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 Alpha systems might be different from the numbers shown in the previous example.

4.2.12 Startup Command Procedures

As a result of the SYSMAN IO commands described in Section 4.2.10 and the *OpenVMS System Management Utilities Reference Manual*, you might need to modify some of your existing SYS$STARTUP:*.COM procedures *if you copy them to an OpenVMS Alpha system disk.* Note, however, that the command procedures provided by OpenVMS Alpha 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.

4.2.13 Devices on OpenVMS Alpha

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

4.2.14 Local DSA Device Naming

On OpenVMS Alpha, 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 Alpha 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 4–7 compares the new OpenVMS Alpha local DSA device-naming scheme with the local naming schemes on OpenVMS VAX and the DEC 7000 Model 600 Alpha console. Note that the DEC 7000 Model 600 Alpha 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 Alpha console commands BOOT and SHOW DEVICE.

Table 4–7 Comparison of Device Naming on OpenVMS

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

As shown in Table 4–7, OpenVMS VAX names disk unit 14 on controller PUB0 as DUB14, while OpenVMS Alpha names this unit DUA14. On OpenVMS Alpha, 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 may preclude 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 Alpha 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 Alpha systems with local controllers 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 Alpha 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 Alpha 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 Alpha. Local DSA physical device names that are hardcoded in command files or applications may also be affected by this change.

4.2.15 File Name Format of Drivers Supplied by Digital on Alpha

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

facility-name$xxDRIVER.EXE

The drivers included on the OpenVMS Alpha 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.

On OpenVMS Alpha nodes, drivers not supplied by Digital still can be called *xx*DRIVER.EXE.

4.2.16 OpenVMS Installation Media

OpenVMS Alpha operating system binaries and documentation are distributed only on a compact disc. Other media for installations are not available. See Section 2.2 for information about providing users access to the online documentation.

4.2.17 VMSINSTAL Utility

The VMSINSTAL utility on OpenVMS Alpha systems includes features that are not available with VMSINSTAL on VAX systems. This section summarizes those VMSINSTAL features.

Note

The OpenVMS VAX and OpenVMS Alpha installations use the POLYCENTER Software Installation utility. This product performs rapid installations and deinstallations, and lets you find information about installed products. These features are more sophisticated than the VMSINSTAL features described in this section. The key points are:

- The POLYCENTER Software Installation utility is the official replacement technology for VMSINSTAL.
- The POLYCENTER Software Installation utility is the preferred technology for packaging and shipping all layered product kits.
- VMSINSTAL will continue to ship on Alpha and VAX systems for compatibility reasons.

- VMSINSTAL is no longer under active development.

See Section 4.2.2 for a summary of the POLYCENTER Software Installation utility features.

The VMSINSTAL utility on OpenVMS Alpha systems contains callbacks that are not available with the VAX version of VMSINSTAL. Software developers at your site who are creating OpenVMS based software kits can read the *OpenVMS Developer's Guide to VMSINSTAL* for details.

4.2.17.1 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.

4.2.17.2 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.

4.2.17.3 Procedure for Listing Installed Products

The SYS$UPDATE:INSTALLED_PRDS.COM procedure 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*
```

4.2.18 Running AUTOGEN

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

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

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 7.1 for information about system parameters and their units and about the tuning considerations.

4.2.19 Installing Main Images and Shareable Images in GHRs

On OpenVMS Alpha, you can improve the performance of main images and shareable images that have been linked with /SHARE and the LINK qualifier /SECTION_BINDING=(CODE,DATA) by installing them as resident with the Install utility (INSTALL). The code and read-only data sections of an installed resident image can reside in granularity hint regions (GHRs) in memory. The Alpha 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 an improvement in TB hit rates, resulting in higher performance.

Also, the OpenVMS Alpha 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 Alpha 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 7.5 for details.

4.2.20 SYS.EXE Renamed to SYS$BASE_IMAGE.EXE

On OpenVMS Alpha systems, the loadable executive image (SYS.EXE on VAX) has been renamed SYS$BASE_IMAGE.EXE. The file resides on the system disk in SYS$LOADABLE_IMAGES.

4.2.21 Rounding-Up Algorithm for Input Quota Values

Be careful when you assign and read the OpenVMS Alpha SYSUAF process quotas that have values in pagelets (WSDEFAULT, WSQUOTA, WSEXTENT, and PGFLQUOTA). OpenVMS Alpha utilities accept and display these quota values in pagelets, and then round up (if warranted). Rounding up occurs on an Alpha 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 Alpha computer with 8KB pages, 2100 pagelets equal 131.25 Alpha pages. The result is that Alpha rounds up to 132 Alpha pages. Thus, specifying 2100 pagelets is effectively the same as specifying a value in the range of 2096 to 2112 pagelets.

The Alpha 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 Alpha 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 Alpha pagelets, and you increase the value by 10 pagelets to 1995 pagelets. On an Alpha computer with 8KB pages, 1985 pagelets equals 124.0625 Alpha pages, which is rounded up internally to 125 Alpha pages. The new, higher value of 1995 pagelets equals 124.6875 Alpha pages, which results in the same 125 Alpha 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 Alpha computer with 8KB pages, 50000 pagelets equals 3125 Alpha 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 Alpha computer with 8KB pages, the 50010 pagelets equals 3125.625 Alpha pages, which is rounded up to 3126 Alpha pages. The 3126 Alpha 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 Alpha page boundaries are crossed would be even greater on Alpha computers with 16KB, 32KB, or 64KB pages.

4.2.22 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 Alpha TFF differs from the OpenVMS VAX TFF in the following ways:

- On OpenVMS Alpha, the TFF fallback driver is SYS$FBDRIVER.EXE. On OpenVMS VAX, the TFF fallback driver is FBDRIVER.EXE.
- On OpenVMS Alpha, the TFF startup file is TFF$*SY*STARTUP.COM. On OpenVMS VAX, the TFF startup file is TFF$STARTUP.COM.
- On OpenVMS Alpha, TFF can handle 16-bit character fallback. The fallback table library (TFF$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 16-bit character fallback.
- On OpenVMS Alpha, 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.

5 Maintenance Tasks

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

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

5.1 Maintenance Tasks That Are the Same

Table 5–1 lists the OpenVMS system management maintenance tasks that are identical or similar on Alpha and VAX.

Table 5–1 Identical or Similar OpenVMS 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 Alpha.
ALLOCATE command	Identical.
MOUNT command	Identical.
Accounting utility commands	Identical.
Analyzing error logs	The ANALYZE/ERROR_LOG command is the same on VAX and Alpha systems, with one exception: the /SUMMARY qualifier is not supported on OpenVMS Alpha systems. On OpenVMS Alpha, 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 Alpha 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 Alpha .OBJ and .EXE files.
ANALYZE/PROCESS_DUMP command	Identical.
Other ANALYZE commands, /AUDIT, /CRASH_DUMP, /DISK_STRUCTURE, /MEDIA, /RMS_FILE, /SYSTEM	Identical.

(continued on next page)

Table 5–1 (Cont.) Identical or Similar OpenVMS Maintenance Tasks

Feature, Task, or Command	Comments
Backing up data	The BACKUP command and its qualifiers are the same. Important: Thoroughly read the *OpenVMS AXP Version 6.1 Release Notes* for the latest information about any restrictions with backup and restore operations on Alpha and VAX systems.
Batch and print queuing system	The same as with OpenVMS VAX Version 6.1. Includes the multiple queue managers feature. See Section 5.2.2 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	Identical. Note that VAX VMS Version 5.5 and earlier releases included the NONPAGED POOL STATISTICS class with the MONITOR ALL_CLASSES command. However, OpenVMS VAX Version 6.0 and later releases, plus OpenVMS Alpha, do not include the NONPAGED POOL STATISTICS class because the former MONITOR POOL feature is not supported. This feature is replaced by adaptive pool management and two SDA commands: SHOW POOL/RING_BUFFER and SHOW POOL/STATISTICS, which are all part of recent OpenVMS releases. See Section 7.4 for more information on adaptive pool management.
MONITOR POOL command	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 Alpha or on OpenVMS VAX Version 6.0 and later releases. This is due to adaptive pool management features and two SDA commands: SHOW POOL/RING_BUFFER and SHOW POOL/STATISTICS. See Section 7.4 for more information on adaptive pool management.
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 Alpha 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 Alpha 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 Alpha processor, where vectors are not supported. On an Alpha computer, MONITOR VECTOR operates the same as on a VAX computer without vector processing.

(continued on next page)

Table 5–1 (Cont.) Identical or Similar OpenVMS Maintenance Tasks

Feature, Task, or Command	Comments
Other MONITOR commands	The following commands are the same: 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.
Defragmenting disks	The OpenVMS **movefile** subfunction, which lets programmers write atomic-file disk-defragmentation applications, is supported on OpenVMS VAX Version 5.5 and later releases, and on OpenVMS Alpha Version 6.1 and later releases. The DEC File Optimizer for OpenVMS layered product, which defragments disks using the **movefile** subfunction, also is supported on OpenVMS VAX and on OpenVMS Alpha.
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 Alpha SYSMAN utility. See Section 4.2.10 and the *OpenVMS System Management Utilities Reference Manual* 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 Alpha systems. (Note: the .STB files are in SYS$LOADABLE_IMAGES and not in SYS$SYSTEM.) System dump file size requirements are higher on OpenVMS Alpha systems. Also, there are new Crash Log Utility Extractor (CLUE) commands on OpenVMS Alpha, and CLUE is part of SDA. See Section 5.2.3 for information about SDA differences.

5.2 Maintenance Tasks That Are Different

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

- OpenVMS Alpha includes an event management utility that is not available on OpenVMS VAX systems. See Section 5.2.1.
- The batch and print queuing system for OpenVMS AXP Versions 6.1 and later is identical to the clusterwide batch and print queuing system in OpenVMS VAX Versions 6.1 and later. See Section 5.2.2.
- Larger system dump files occur and additional values for the DUMPSTYLE system parameter are provided on OpenVMS Alpha. SDA is automatically invoked (after a system crash) when you reboot the system. Also, there are new Crash Log Utility Extractor (CLUE) commands on OpenVMS Alpha, and CLUE is part of SDA. See Section 5.2.3.
- No Patch utility is supplied for OpenVMS Alpha systems. See Section 5.2.4.

5.2.1 DECevent Event Management Utility

The DECevent utility is an event management utility for the OpenVMS Alpha operating system. DECevent provides the interface between a system user and the system's event log files. The utility is invoked by entering the DCL command DIAGNOSE and lets system users create ASCII reports derived from system event entries. The format of the ASCII reports depends on the command entered on the command line interface (CLI) with a maximum character limit of 255 characters.

Event report information can be filtered by event type, date, time, and event entry number. Event report formats from full disclosure to brief informational messages can be selected. The /INCLUDE and /EXCLUDE qualifiers provide a wide range of selection criteria to narrow down the focus of event searches.

The DECevent utility also offers an interactive command shell interface that recognizes the same commands used at the command line interface. From the interactive command shell, users can customize, change, or save system settings.

DECevent uses the system event log file, SYS$ERRORLOG:ERRLOG.SYS, as the default input file for event reporting, unless another file is specified.

The DECevent event management utility provides the following report types:

- Full
- Brief
- Terse
- Summary
- FSTERR

Used with qualifiers, these report types allow a system user to view event information in several ways.

See the *OpenVMS System Manager's Manual* for further details.

5.2.2 Comparison of Batch and Print Queuing Systems

Table 5–2 compares the batch and print queuing systems for recent releases of the operating systems.

Table 5–2 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 or later and OpenVMS Alpha Version 6.1 or later
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 VMScluster nodes (to work around CPU or memory resource shortages).

(continued on next page)

Table 5–2 (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 or later and OpenVMS Alpha Version 6.1 or later
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 VMScluster.
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.
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.

(continued on next page)

Table 5–2 (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 or later and OpenVMS Alpha Version 6.1 or later
/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.
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. Note that the security features in OpenVMS VAX Version 6.0 are certified as C2 compliant by the U.S. Government. OpenVMS VAX Version 6.1 has also been certified as C2 compliant, and OpenVMS AXP Version 6.1 is currently being reviewed.

Multiple queue managers are useful in VMScluster environments with CPU or memory shortages. They allow you to distribute the batch and print work load across different nodes in the cluster. For example, if you create separate queue managers for batch queues and print queues, you can run the batch queue manager on one node and the print queue manager on a different node.

This feature is available with the following restrictions:

- The multiple queue manager feature cannot be used in a VMScluster until one of the following is true:
  - All the nodes are running OpenVMS Version 6.1.
  - The nodes are running OpenVMS AXP Version 6.1 and OpenVMS VAX Version 6.0.
- Once you begin using multiple queue managers, you cannot bring a node running a version earlier than OpenVMS AXP Version 6.1 or OpenVMS VAX Version 6.0 into the VMScluster environment. Doing so might result in unexpected results and failures.
- Queues running on one queue manager cannot reference queues running on a different queue manager. For example, a generic queue running on queue manager A cannot direct jobs to an execution queue running on queue manager B. In addition, you cannot move a job from a queue on queue manager A to a queue on queue manager B.
- No more than five queue managers are allowed in a VMScluster environment.

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

- *OpenVMS System Manager's Manual*
- *OpenVMS DCL Dictionary*

5.2.3 System Dump Analyzer

Differences in the System Dump Analyzer (SDA) occur with the size of the system dump file. See Section 5.2.3.1 for more information; see Section 5.2.3.2 for a related discussion about conserving dump file space. SDA is automatically invoked (after a system crash) when you reboot the system, as discussed in Section 5.2.3.3. Also, there are new Crash Log Utility Extractor (CLUE) commands on Alpha systems, and CLUE is run in SDA; see Section 5.2.3.4.

5.2.3.1 Size of the System Dump File

The location and the file name of the system dump file are the same on OpenVMS VAX and OpenVMS Alpha systems. However, VAX and Alpha 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 Alpha 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)
```

5.2.3.2 Conserving Dump File Storage Space

Dump file storage space might be at a premium on OpenVMS Alpha 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 Alpha systems. When a system fails on a VAX or Alpha 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 Alpha computers might not have consoles, in which case this console data never appears.

VAX systems always have full console output. On Alpha 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 Alpha includes options for shorter versions of the console output. Digital picked the values 0 and 1 for the shorter console output on OpenVMS Alpha systems so that you do not have to change your DUMPSTYLE system parameter to get the default, shorter output.

Table 5–3 compares the values for the DUMPSTYLE parameter.

Table 5–3 Comparison of DUMPSTYLE System Parameter Values

Value	Meaning on OpenVMS VAX	Meaning on OpenVMS Alpha
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 Alpha and VAX systems, a SHOW command in SYSGEN lists the default value for the DUMPSTYLE system parameter as 0. However, on OpenVMS Alpha, 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.

5.2.3.3 SDA Automatically Invoked at System Startup

On OpenVMS Alpha systems, the System Dump Analyzer (SDA) is automatically invoked (after a system crash) when you reboot the system. To facilitate crash dump analysis, the SDA Crash Log Utility Extractor (CLUE) automatically captures and archives selective dump file information in a CLUE list file.

A startup command procedure initiates commands that:

- Invoke SDA
- Create a CLUE$*node_ddmmyy_hhmm*.LIS file
- Issue a CLUE HISTORY command to populate this .LIS file

If these files accumulate more than 5000 blocks of storage, the oldest file is deleted. The contents of each file is where most analysis of a system failure begins. To inhibit the running of CLUE at system startup, either rename the SYS$STARTUP:CLUE$SDA_STARTUP.COM file, or in the SYLOGICALS.COM file define CLUE$INHIBIT as /SYS TRUE.

5.2.3.4 Using SDA CLUE Commands to Obtain and Analyze Crash Dump Information

On Alpha systems, SDA Crash Log Utility Extractor (CLUE) extension commands can summarize information provided by certain standard SDA commands and also provide additional detail for some SDA commands. These SDA CLUE commands can interpret the contents of the dump to perform additional analysis.

The CLUE commands are:

```
CLUE CLEANUP
CLUE CONFIG
CLUE CRASH
CLUE DEBUG
CLUE ERRLOG
CLUE HISTORY
CLUE MCHK
CLUE MEMORY
CLUE PROCESS
CLUE STACK
CLUE VCC
CLUE XQP
```

You can use all CLUE commands when analyzing crash dumps; the only CLUE commands that are not allowed when analyzing a running system are CLUE CRASH, CLUE ERRLOG, CLUE HISTORY, and CLUE STACK.

Understanding SDA CLUE

When rebooting after a system failure, SDA is automatically invoked, and SDA CLUE commands analyze and save summary information from the crash dump file in CLUE history and listing files.

The CLUE HISTORY command updates the history file pointed to by the logical name CLUE$HISTORY with a one-line entry and the major crash dump summary information.

The CLUE listing file contains more detailed information about the system crash and is created in the directory pointed to by the logical name CLUE$COLLECT. The following information is included in the CLUE listing file:

- Crash dump summary information
- System configuration
- Stack decoder
- Page and swap files
- Memory management statistics
- Process DCL recall buffer
- Active XQP processes
- XQP cache header

You can print the CLUE list file immediately or save it for later examination.

5.2.4 Patch Utility Not Supported

The OpenVMS VAX Patch utility (PATCH) is not supported on OpenVMS Alpha because compiler performance optimizations and the Alpha architecture organize the placement of instructions and data in an image in a way that makes patching impractical.

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 Alpha systems.

6 Security Tasks

The security features in OpenVMS Alpha are nearly identical to those found in OpenVMS VAX. Note the following exceptions:

- The C2 features in OpenVMS VAX Version 6.0 were formally evaluated and certified by the U.S. Government. OpenVMS VAX Version 6.1 has also been certified as C2 compliant, and OpenVMS AXP Version 6.1 is currently being reviewed.
- OpenVMS Alpha does not audit DECnet network connections.

See the *OpenVMS Guide to System Security* for details about the OpenVMS security features that are common on Alpha and VAX environments.

7 Performance Optimization Tasks on Alpha Systems

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

- Impact of the larger Alpha page size on system parameter units (see Section 7.1).
- Changes to the default values for a number of system parameters (see Section 7.2).
- Use of page or pagelet values by OpenVMS utilities and DCL commands (see Section 7.3).
- Adaptive pool management feature (see Section 7.4).
- Installation of suitably linked main images and shareable images in a granularity hint region for improved performance (see Section 7.5).
- The virtual I/O cache (also part of OpenVMS VAX Version 6.0 and later releases) that reduces bottlenecks and improves performance (see Section 7.6).

See the *Guide to OpenVMS AXP Performance Management* for more information on OpenVMS Alpha performance considerations.

7.1 System Parameters: Measurement Change for Larger Page Size

As discussed in Section 3.2.1 and as illustrated in Figure 3–1, a VAX page is 512 bytes, and an Alpha page can be 8KB, 16KB, 32KB, or 64KB. The initial set of Alpha computers use a page size of 8KB (8192 bytes).

The larger page size for an Alpha system required 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 Alpha.

The OpenVMS VAX term *page* is ambiguous in the following ways for certain system parameters in the Alpha 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 Alpha users.

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

- Units that have changed in name only (see Section 7.1.1).
- Units that are CPU specific (see Section 7.1.2).
- Parameters that have both external and internal values (see Section 7.1.3).

7.1.1 System Parameter Units That Changed in Name Only

Table 7–1 shows the system parameters whose units have changed in name only, from “page” on OpenVMS VAX to a new, more appropriate name on OpenVMS Alpha.

Table 7–1 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

7.1.2 CPU-Specific System Parameter Units

Table 7–2 shows the units that remain as CPU-specific pages (8KB on the initial set of Alpha computers).

Table 7–2 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

7.1.3 System Parameters with Dual Values

Table 7–3 shows the parameter units that have both an external value and an internal value on OpenVMS Alpha.

Table 7–3 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 Alpha 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 7–3 (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 Alpha:

```
Parameter Name            Current    Default     Min.     Max.    Unit  Dynamic
--------------            -------    -------    -------  -------  ----  -------
PFCDEFAULT                    128        128        0      2032  Pagelets   D
 internal value                 8          8        0       127  Pages      D
GBLPAGES                   443040      30720    10240        -1  Pagelets
 internal value             27690       1920      640        -1  Pages
SYSMWCNT                     5288       2048      512     65536  Pagelets
 internal value               331        128       32      4096  Pages
WSMAX                       32768       4096     1024   1048576  Pagelets
 internal value              2048        256       64     65536  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                        512        512        0        -1  Pagelets   D
 internal value                32         32        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 Alpha 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 Alpha 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 Alpha 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 4.2.21 for a related discussion of the rounding-up process with process quotas. Also, see Figure 3–1 for an illustration of the relative sizes of a VAX page, an Alpha pagelet, and an Alpha 8KB page.

7.2 Comparison of System Parameter Default Values

Table 7–4 shows the OpenVMS Alpha system parameters whose default values, as noted in SYSGEN, are different from the value on OpenVMS VAX.

Note

Table 7–4 does not repeat the OpenVMS Alpha system parameters listed in Table 7–3 when the only difference is in the name of the unit (a 512-byte VAX page to a 512-byte Alpha pagelet). For example, the PFCDEFAULT default value on a VAX system is 32 pages; its default value on an Alpha system is 32 pagelets, the same quantity.

Also, as you compare the columns, remember:

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

Figure 3–1 illustrates the relative sizes of a VAX page, an Alpha pagelet, and an Alpha 8KB page.

Table 7–4 Comparison of System Parameter Default Values

Parameter	VAX Value	Alpha Value
GBLPAGES	15000 pages	30720 pagelets
GBLPAGFIL	1024 pages	128 pages[1]
SYSMWCNT	508 pages	2048 pagelets
BALSETCNT	16 slots	30 slots
WSMAX	1024 pages	4086 pagelets
NPAGEDYN	620000 bytes	524288 bytes
PAGEDYN	214100 bytes	212992 bytes
VIRTUALPAGECNT	12032 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	512 pagelets
SWPOUTPGCNT	288 pages	512 pagelets
MAXBUF	2064 bytes	8192 bytes

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

(continued on next page)

Table 7–4 (Cont.) Comparison of System Parameter Default Values

Parameter	VAX Value	Alpha Value
CLISYMTBL	250 pages	512 pagelets
LNM*S*HASHTBL	128 entries	512 entries
LNM*P*HASHTBL	128 entries	512 entries
PQL_DBIOLM	18 I/Os	32 I/Os
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	1024 pagelets
PQL_MWSDEFAULT	60 pages	512 pagelets
PQL_DWSQUOTA	200 pages	2048 pagelets
PQL_MWSQUOTA	60 pages	1024 pagelets
PQL_DWSEXTENT	400 pages	16384 pagelets
PQL_MWSEXTENT	60 pages	2048 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 Alpha 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.

7.3 Use of Page or Pagelet Values in Utilities and Commands

Section 3.2.1 describes the relative sizes of a VAX page (512 bytes), an Alpha pagelet (also 512 bytes), and a CPU-specific Alpha page (8192 bytes on an Alpha computer with 8KB pages). Section 7.1 and Section 7.2 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 16-DEC-1994 13:46:41.99

Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (96.00Mb)            12288       10020        2217          51

Virtual I/O Cache (Kbytes):        Total        Free      In Use
  Cache Memory                      3200           0        3200

Granularity Hint Regions (pages):  Total        Free      In Use    Released
  Execlet code region                512           0         271         241
  Execlet data region                128           1          63          64
  VMS exec data region               200           3         197           0
  Resident image code region         512           0         258         254

Slot Usage (slots):                Total        Free    Resident     Swapped
  Process Entry Slots                119         110           9           0
  Balance Set Slots                  117         110           7           0

Dynamic Memory Usage (bytes):      Total        Free      In Use     Largest
  Nonpaged Dynamic Memory        1196032      846080      349952      674496
  Paged Dynamic Memory           1294336      826368      467968      825888

Paging File Usage (blocks):                     Free  Reservable       Total
  DISK$AlphaVMSSYS:[SYS0.SYSEXE]SWAPFILE.SYS     15104       15104       15104
  DISK$AlphaVMSSYS:[SYS0.SYSEXE]PAGEFILE.SYS    204672      184512      204672

Of the physical pages in use, 1789 pages are permanently allocated to VMS.
```

- 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 Alpha systems, the SHOW SYSTEM/FULL command displays CPU-specific pages *and kilobytes* in the rightmost column, Pages. For example:

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

OpenVMS V5.5-2  on node VAXCPU  22-AUG-1994 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 Alpha node and you are displaying process information from the same or another Alpha 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 Alpha pages, the value beneath it would be 1768KB (221*8).

The SHOW SYSTEM command on OpenVMS Alpha displays “OpenVMS” and the version number in the banner and does not display “VAX” or “Alpha.”

- 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-1994 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-1994 09:55:47.37   User: SMART           Process ID:   2D000215
                          Node: SAMPLE          Process name: "SMART"
   .
   .
   .
Process Quotas:
 Account name: DOC
 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 Alpha computer assign pagelets to the appropriate DCL command qualifiers, keep in mind the previously stated caveats—as noted in Section 4.2.21 and Section 7.1—about how pagelet values that are not multiples of 16 (on an Alpha computer with 8KB pages) are rounded up to whole Alpha pages.

7.4 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 any version of OpenVMS Alpha, or OpenVMS VAX Version 6.0 and later releases. The feature is 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 Alpha 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 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 any version of OpenVMS Alpha, or on OpenVMS VAX Version 6.0 and later releases.

7.5 Installing Main Images and Shareable Images In GHRs

On OpenVMS Alpha, you can improve the performance of main images and shareable images that have been linked with /SHARE and the LINK qualifier /SECTION_BINDING=(CODE,DATA) by installing them as resident with the Install utility (INSTALL).

These options are not available on OpenVMS VAX systems.

The code sections and read-only data 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 Alpha 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 an increase in TB hit rates. Consequently, TBs on Alpha systems are used more efficiently than if the loadable executive images or the shareable images were loaded in the traditional manner.

Also, the OpenVMS Alpha 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 system parameters, as discussed in Section 7.5.2.

The GHR feature lets OpenVMS split the contents of images and sort the pieces so 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 the Install utility and SYSGEN 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.

The DATA keyword indicates that the linker must ensure that no relative references exist between data image sections. If the image is linked with the DATA option on the /SECTION_BINDING qualifier, the image activator compresses the data image sections in order not to waste virtual address space. Although it does save virtual address space, the DATA option to the /SECTION_ BINDING qualifier does not improve performance.

See the *OpenVMS Linker Utility Manual* for details about the /SECTION_ BINDING qualifier.

You can use the ANALYZE/IMAGE command on an OpenVMS Alpha 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.

7.5.1 Install Utility Support

Several OpenVMS Alpha 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.

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$SHR.EXE, DECC$SHR.EXE, and CMA$TIS_SHR.EXE images are installed resident on OpenVMS Alpha.

7.5.2 System Parameter Support

Five system parameters are associated with the GHR feature:

- GH_RSRVPGCNT
- GH_EXEC_CODE
- GH_EXEC_DATA
- GH_RES_CODE
- GH_RES_DATA

Table 7–5 summarizes the purpose of each system parameter.

Table 7–5 System Parameters Associated with GHR Feature

System Parameter	Description
GH_RSRVPGCNT	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.
GH_EXEC_CODE	Specifies the size in pages of the execlet code granularity hint region.
GH_EXEC_DATA	Specifies the size in pages of the execlet data granularity hint region.
GH_RES_CODE	Specifies the size in pages of the resident image code granularity hint region.
GH_RES_DATA	Specifies the size in pages of the resident image data granularity hint region.

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

7.5.3 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.

7.5.4 Loader Changes for Executive Images in GHRs

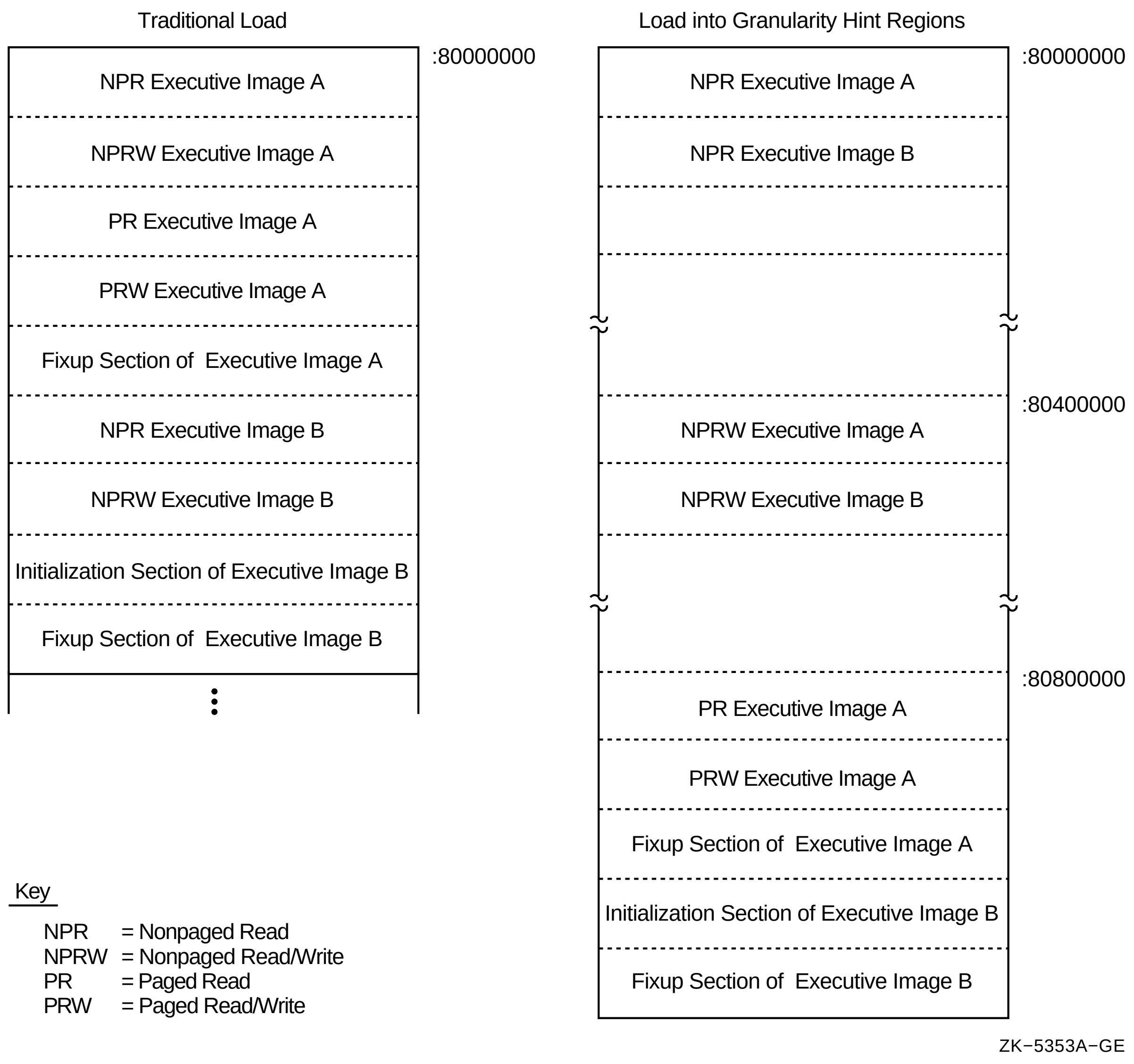
In traditional executive image loading, code and data are sparsely laid out in system address space. The loader allocates the virtual address space for executive images so that the image sections are loaded on the same boundaries created for them by the linker.

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 7–1 illustrates a traditional load and a load into a GHR.

Figure 7–1 Traditional Loads and Loads into GHRs



7.6 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 functionality provides a write-through cache that maintains the integrity of disk writes while significantly improving read performance.

Virtual I/O caching takes place if all members of the cluster have caching enabled. If one member does not have caching enabled, perhaps because it is running OpenVMS Version 5.5-2, then virtual I/O caching will not occur.

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.

8
Network Management Tasks

You can use DECnet for OpenVMS Alpha to establish networking connections with other nodes. DECnet for OpenVMS, previously known as DECnet–VAX on the VAX/VMS platform, implements Phase IV of the Digital Network Architecture. The DECnet for OpenVMS Alpha features are similar to those of the DECnet for OpenVMS VAX, with a few exceptions. This chapter:

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

Note

The comparisons in this chapter are specific to the Phase IV software, DECnet for OpenVMS. If your OpenVMS Alpha node is using DECnet/OSI for OpenVMS (Phase V), refer to the DECnet/OSI manuals and release notes for details.

If you install the version of DECnet/OSI that is for OpenVMS Alpha:

- DECdns client is available.
- X.25 support is available through DEC X.25 for OpenVMS Alpha. The QIO interface drivers from the P.S.I. product are included in DEC X.25 in order to provide the same interface to customer applications.

8.1 Network Management Tasks That Are the Same

Table 8–1 lists the OpenVMS network management tasks that are identical or similar on Alpha and VAX computers.

Table 8–1 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 Alpha and VAX systems. However, the PAK name enabling cluster alias routing support on OpenVMS Alpha (DVNETEXT) is different from the PAK name enabling cluster alias routing support on OpenVMS VAX (DVNETRTG). See Section 8.2.1 for related information.

(continued on next page)

Table 8–1 (Cont.) Identical or Similar OpenVMS Network Management Tasks

Feature or Task	Comment
Cluster alias	Similar; however, level 1 routing is supported only on routers for a cluster alias. See Section 8.2.1 for more information.
NETCONFIG.COM procedure	The same, with one exception. See Section 8.2.1 for more information.
NETCONFIG_UPDATE.COM procedure	Identical.
Configuring DECnet databases and starting OpenVMS Alpha 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 Alpha (with the limitations in Section 8.2 in mind). The functions of the SYS$MANAGER:STARTNET.COM procedure are similar.
File access listener (FAL)	FAL is fully compatible with the OpenVMS VAX version. For example, bidirectional file transfer using 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). Note, however, that the size of the network may be much larger if you are running DECnet/OSI for OpenVMS Alpha. See your DECnet/OSI documentation for details.
Node name rules	The same rules and 6-character maximum length as with VAX nodes running DECnet for OpenVMS.
DECnet objects	Identical.
Task-to-task communication	Identical.
Network management using 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 Alpha. 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 8.2.5 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 only.
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 8.2.1 and Section 8.2.5 for details.
SET HOST capabilities	Identical.

8.2 Network Management Tasks That Are Different

This section describes the OpenVMS network management tasks that are different on Alpha 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 (see Section 8.2.1).
- Some line types are not supported (see Section 8.2.2).
- The Distributed Name Service (DNS) node name interface that is used on OpenVMS VAX is not supported on OpenVMS Alpha (see Section 8.2.3).
- VAX P.S.I. is not supported (see Section 8.2.4).
- A number of NCP command parameters are affected by unsupported features (see Section 8.2.5).

8.2.1 Level 1 Routing Supported for Cluster Alias Only

Level 1 DECnet routing is available, but is supported only on DECnet for OpenVMS Alpha 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 Alpha nodes.

Note that the Product Authorization Key (PAK) name enabling DECnet for OpenVMS Alpha 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 Alpha 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 Alpha and VAX systems is that you have to manually enable level 1 routing on one of the Alpha 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 ! Only neccessary on cluster alias routers
NCP>DEFINE NODE alias-node-address NAME alias-node-name
NCP>DEFINE EXECUTOR ALIAS NODE alias-node-name
NCP>DEFINE EXECUTOR ALIAS INCOMING {ENABLED DISABLED}
```

[1] In a VMScluster, you do not have to enable level 1 routing on an Alpha node if one of the VAX VMS Version 5.5–2 nodes or OpenVMS VAX Version 6.*n* nodes is a routing node.

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 that has this parameter set to ENABLED will handle incoming alias connections.

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.

See Section 8.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS Alpha systems because level 2 routing is not supported and level 1 routing is reserved for the cluster alias.

8.2.2 CI and DDCMP Lines Not Supported

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

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 Alpha systems.

See Section 8.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS Alpha systems because DDCMP is not supported.

8.2.3 DNS Node Name Interface Not Supported

With DECnet for OpenVMS Alpha (Phase IV), the Distributed Name Service (DNS) node name interface that is used on OpenVMS VAX is not supported. Consequently, DNS object names cannot be specified by users and applications on OpenVMS Alpha nodes running Phase IV. See Section 8.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS Alpha systems because DNS is not supported.

Note that the DECdns client is supported with DECnet/OSI. See the DECnet/OSI documentation for details.

8.2.4 VAX P.S.I. Not Supported

VAX P.S.I., a software product that enables connections to X.25 networks, is not supported. See Section 8.2.5 for information about NCP command parameters that are available but have no effect on OpenVMS Alpha systems because VAX P.S.I. is not supported.

Although VAX P.S.I. is not supported, a DECnet for OpenVMS Alpha node can communicate with DECnet nodes that are connected to X.25 networks reachable via a DECnet/X.25 router.

8.2.5 NCP Command Parameters Affected by Unsupported Features

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

- DDCMP.
- VAX P.S.I. (However, a DECnet for OpenVMS Alpha node can communicate with DECnet nodes that are connected to X.25 networks reachable via a DECnet/X.25 router.)
- 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 OpenVMS Alpha characteristic 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 8–2 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 8–2 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

(continued on next page)

Table 8–2 (Cont.) NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
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
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.

[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 8–2 (Cont.) NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
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.
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.

(continued on next page)

Table 8–2 (Cont.) NCP Command Parameters Affected by Unsupported Features

NCP Command Parameter	Associated Unsupported Feature
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 only supported for cluster alias routers.
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.
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.

9 Interoperability of OpenVMS VAX and OpenVMS Alpha

This chapter describes the similarities and differences, where they exist, of the following topics:

- Interoperability of OpenVMS VAX and OpenVMS Alpha on a DECnet network and on a TCP/IP network
- DECnet (Phase IV) network features and management
- DECnet/OSI network features
- Interoperability of OpenVMS VAX and OpenVMS Alpha in a VMScluster

9.1 Interoperability on a Network

OpenVMS Alpha and OpenVMS VAX systems can interoperate in a network, sharing system resources and enabling communication with local and remote nodes, in the same way that OpenVMS VAX systems interoperate. Network management of OpenVMS Alpha nodes is also similar to network management of OpenVMS VAX nodes. However, some differences exist due to architectural and implementation differences.

9.1.1 Interoperability Using DECnet for OpenVMS and DECnet/OSI for OpenVMS

DECnet for OpenVMS and DECnet/OSI for OpenVMS are used to establish networking connections with other OpenVMS VAX and OpenVMS Alpha nodes. Both products are available for OpenVMS VAX systems and for OpenVMS Alpha systems.

DECnet for OpenVMS, formerly known as DECnet–VAX, implements Phase IV of the Digital Network Architecture (DNA). The features of DECnet for OpenVMS Alpha are similar to those of the DECnet–VAX software that is part of VMS Version 5.4–3, with a few exceptions, as noted in Section 9.2. This product is included with the OpenVMS operating system and ships on the OpenVMS compact disc.

DECnet/OSI for OpenVMS, which implements Phase V of DNA, is an ISO-compliant product. DECnet/OSI for OpenVMS conforms to the Open Systems Interconnection (OSI) networking standards defined by the International Organization for Standardization (ISO). DECnet/OSI provides all the features of DECnet Phase IV as well as OSI features that enable participation in open standards based networks. DECnet/OSI is a layered product on OpenVMS and ships on the layered product compact disc.

Full names support for DECnet/OSI was introduced in OpenVMS VAX Version 6.1 and is now available on both OpenVMS VAX and OpenVMS Alpha.

You can transfer files over the DECnet network, including copying and printing, between OpenVMS VAX and OpenVMS Alpha systems running either DECnet for OpenVMS or DECnet/OSI for OpenVMS.

The SET HOST command enables full interoperability for DECnet remote login between OpenVMS VAX and OpenVMS Alpha nodes running DECnet for OpenVMS or DECnet/OSI for OpenVMS.

9.1.2 Interoperability Using TCP/IP Networking on OpenVMS Systems

TCP/IP software supports communication between computer systems of similar or different design as well as interconnection of various physical networks to form larger networks. There are several vendors of TCP/IP software for OpenVMS (see Appendix A of *TCP/IP Networking on OpenVMS Systems*). Users on an OpenVMS system running TCP/IP software can execute the basic applications shown in Table 9–1.

Table 9–1 Applications of TCP/IP Software for OpenVMS

Application Type	Vendor Specific[1]	Vendor Common[2]
Virtual terminal service	RLOGIN	SET HOST/RLOGIN
	TELNET	SET HOST/TELNET
File access	FTP	COPY/FTP, DIR/FTP
	RCP	COPY/RCP
Disk service	NFS[3]	

[1]Same commands are available but qualifiers and parameters may differ.

[2]Commands, qualifiers, and parameters are identical for all named vendors.

[3]Not all implementations support the Network File System (NFS).

The TCP/IP software can also be used to connect to and access the global Internet. For more information about TCP/IP networking on OpenVMS, see *TCP/IP Networking on OpenVMS Systems*.

9.1.3 Network Interfaces

Most of the network protocols, buses, and interconnects that are supported on OpenVMS VAX systems are also supported on OpenVMS Alpha systems, as shown in the following tables.

9.1.3.1 Network Protocols

The network protocols supported on OpenVMS VAX and OpenVMS Alpha systems are shown in Table 9–2.

Table 9–2 Network Protocol Support

Protocol	OpenVMS VAX Versions V5.5-2 and later	OpenVMS Alpha Versions 1.5 and later
DECnet (Phase IV)	Yes	Yes
DECnet/OSI (Phase V)	Yes	Yes
LAD/LAST	Yes	Yes
LAT	Yes	Yes
LAVC	Yes	Yes
TCP/IP[1]	Yes	Yes
X.25	Yes[2]	Yes[3]

[1] Provided by one of the TCP/IP for OpenVMS vendors listed in Appendix A of *TCP/IP Networking on OpenVMS Systems*.

[2] For OpenVMS VAX Version V5.5-2, provided by VAX P.S.I.; for OpenVMS VAX Version 6.0 and later, provided by DECnet/OSI.

[3] For OpenVMS Alpha Version 1.5 and Version 1.5-1H1, provided by DEC X.25 Client for OpenVMS Alpha software. Running this software, a node can connect to an X.25 network via an X.25 gateway. For systems running DECnet/OSI, X.25 support is provided by X.25 for OpenVMS Alpha, which includes both client and native X.25 functionality.

9.1.3.2 Buses

The buses supported on OpenVMS VAX and OpenVMS Alpha systems are shown in Table 9–3. Support is also dependent on the computer model.

Table 9–3 Bus Support

Bus	OpenVMS VAX Versions V5.5-2 and later	OpenVMS Alpha Versions 1.5 and later
BI-bus	Yes	No[1]
DSSI	Yes	Yes
EISA bus	No	Yes[2]
Futurebus+	No	Yes[2]
ISA	No	Yes[3]
PCI	No	Yes[4]
Q–bus	Yes	No[1]
SCSI	Yes	Yes
TURBOchannel	Yes	Yes
UNIBUS	Yes	No[1]
VME	Yes	No[5]
XMI	Yes	Yes

[1] Support not planned.

[2] Except for OpenVMS AXP Version 1.5.

[3] OpenVMS AXP Version 6.1-1H1 and later.

[4] OpenVMS AXP Version 6.1 and later.

[5] A custom Digital offering is available.

9.1.3.3 Interconnects

The interconnects (including their respective protocols and drivers) that are supported on OpenVMS VAX and OpenVMS Alpha systems are shown in Table 9–4.

Table 9–4 Interconnect Support

Interconnect	DECnet[1]	TCP/IP[2]	LAT	Cluster: Computer to Computer	Cluster: Computer to Tape or Disk
Asynchronous line	V	–	–	–	–
ATM[3]	A	A	A	A	–
CI	V	–	–	A,V	A,V
DSSI	–	–	–	A,V	A,V
Ethernet	A,V	A,V	A,V	A,V	–
FDDI[4]	A,V	A,V	A,V	A[5],V	–
SCSI	–	–	–	–	A[6],V[7]
Synchronous line	V	–	–	–	–
Token Ring	–	A[8]	–	–	–

[1]Provided by DECnet for OpenVMS and DECnet/OSI for OpenVMS.

[2]Provided by one of the TCP/IP for OpenVMS vendors listed in Appendix A of *TCP/IP Networking on OpenVMS Systems*.

[3]Available on TurboChannel only, in point-to-point configurations.

[4]Boot support is not available for all computer models (see Table 9–5).

[5]OpenVMS AXP Versions 1.5 and 1.5-1H1 do not use an FDDI adapter for cluster communications, but Ethernet bridging to FDDI backbones can be used. OpenVMS Alpha, starting with Version 6.1 can use an FDDI adapter for cluster communications.

[6]One computer per SCSI bus, serving tape and disk. Two computers on the SCSI bus, with disks only, starting with OpenVMS Alpha Version 6.2.

[7]One computer per SCSI bus, serving tape and disk.

[8]Available only on OpenVMS Alpha, starting with Version 6.1.

Key

A = OpenVMS AXP Version 1.5 and later
V = OpenVMS VAX Version V5.5-2 and later
– = Neither

9.1.3.4 FDDI Boot Support on OpenVMS Alpha

Although FDDI is supported as an interconnect, it does not provide boot support for any VAX computer models. However, it does provide boot support for several Alpha computer models, as shown in Table 9–5.

Note

Configurations without FDDI boot support must also be wired to an Ethernet circuit if either of the following conditions exists:

- Operating system software installations or updates are to be performed from an InfoServer
- System is to be a VMScluster satellite (unlikely for larger systems)

Table 9–5 FDDI Boot Support for OpenVMS Alpha Version 6.2

Interconnect	Adapter	Alpha Computer	Boot Support
EISA			
	DEFEA	1000	Yes
	DEFEA	2000	No
	DEFEA	2100	Yes
XMI			
	DEMFA	7000	Yes
	DEMFA	8200	Yes
	DEMFA	8400	Yes
Futurebus+			
	DEFAA	4000	No
	DEFAA	7000	No
TURBOchannel			
	DEFZA	3000	No
	DEFTA	3000	Yes
PCI			
	DEFPA	200	No
	DEFPA	250	No
	DEFPA	400	No
	DEFPA	600	No
	DEFPA	1000	No
	DEFPA	2000	No
	DEFPA	2100	No
	DEFPA	8200	No
	DEFPA	8400	No

9.2 DECnet (Phase IV) Network Features and Management

The similarities and differences in the network features and management for DECnet for OpenVMS Alpha (Phase IV) and for DECnet–VAX for VMS Version 5.4–3 are described in this section.

9.2.1 Similarities

The features and management of DECnet for OpenVMS Alpha (Phase IV) are similar to those of DECnet–VAX for VMS Version 5.4–3, with some exceptions. The following list shows the features and management tasks that are identical:

- DECnet objects
- DECnet Test Sender/DECnet Test Receiver utility (DTS/DTR)
- Downline load and upline dump operations
- Event logging
- Ethernet monitor (NICONFIG)
- File access listener (FAL)

  (Identical between Alpha nodes and between Alpha and VAX nodes.)
- Loopback mirror testing
- NETCONFIG_UPDATE.COM procedure
- Node name rules
- Product Authorization Key (PAK) name for end-node license (DVNETEND)
- SET HOST capabilities

  (Identical between Alpha nodes and between Alpha and VAX nodes.)
- Size of network
- Task-to-task communication

9.2.2 Differences

The features and management tasks of DECnet for OpenVMS Alpha (Phase IV) are similar to those of DECnet–VAX for VMS Version 5.4–3, with some exceptions, as shown in Table 9–6.

Table 9–6 Differences of DECnet Features and Management Tasks

Feature or Task	OpenVMS VAX	OpenVMS Alpha
Cluster alias	Level 1 and Level 2 routing supported on nodes acting as routers for a cluster alias.	Only Level 1 routing supported on nodes acting as routers for a cluster alias.

(continued on next page)

Table 9–6 (Cont.) Differences of DECnet Features and Management Tasks

Feature or Task	OpenVMS VAX	OpenVMS Alpha
Configuring DECnet databases and starting OpenVMS Alpha computer's access to the network		Process for both is the same but subject to the routing limitations and the lack of support for CI, DDCMP, the Distributed Name Service (DNS) node name interface, VAX P.S.I., and certain Network Control Program (NCP) utility command parameters. The functions of the SYS$MANAGER:STARTNET.COM procedure are similar.
Lines supported	CI, asynch (DDCMP), Ethernet, and FDDI	Ethernet and FDDI
NETCONFIG.COM procedure (part for specifying a router)	NETCONFIG.COM prompts you, "Do you want to operate as a router?"	NETCONFIG.COM does not prompt you. You have to enable Level 1 routing manually. Otherwise, NETCONFIG.COM is the same on OpenVMS Alpha systems.
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, the NCP command parameters for SET and SHOW operations for DDCMP, full host-based routing, the Distributed Name Service (DNS), and VAX P.S.I. have no effect on OpenVMS Alpha.
Product Authorization Key (PAK) name for cluster alias routing support	DVNETRTG	DVNETEXT
Routing	Level 1 and Level 2 routing are supported.	Level 1 routing is supported only on nodes acting as routers for a cluster alias. Level 2 routing is not supported.
VAX P.S.I.	Supported (except DEC 2000)	Not supported. For DECnet for OpenVMS Alpha Version 1.5, DEC X.25 Client for OpenVMS Alpha replaces VAX P.S.I. For DECnet/OSI, X.25 for OpenVMS Alpha provides both client and native X.25 functionality.

For more information about the system management differences between DECnet for OpenVMS (Phase IV) on OpenVMS VAX and OpenVMS Alpha systems, see Chapter 8.

9.3 DECnet/OSI for OpenVMS Network Features

Digital's open DECnet/OSI for OpenVMS network is a family of hardware and software products that allows Digital operating systems to communicate with each other and with systems produced by other vendors.

The DECnet/OSI network includes the following features:

- Remote system communication
- Resource sharing
- Support for distributed processing

- Distributed network management
- Optional use of distributed system services for networkwide names and synchronized time

Network users can access resources on any system in the network and the resources of other vendors' systems on multivendor networks.

DECnet/OSI for OpenVMS is Digital's implementation for OpenVMS systems of:

- The Open Systems Interconnection (OSI) communications specifications, as defined by the International Standards Organization (ISO).
- Digital's communications architecture, Digital Network Architecture (DNA) Phase V, which is also backward compatible with the Phase IV architecture.

Phase V integrates the DNA and OSI layers. The DNA Phase V Reference Model is the architectural model on which DECnet/OSI networking implementations are based.

Table 9–7 shows the changes that have evolved with each new phase.

Table 9–7 DNA Phases

Phase	Number of Nodes
I	Limited to 2
II	Up to 32: file transfer, remote file access, task-to-task programming interfaces, network management
III	Up to 255: adaptive routing, downline loading, record access
IV	Up to 64,449: Ethernet local area networks, area routing, host services, VMScluster support
V	Virtually unlimited number of systems: OSI protocol support, transparent transport level links to TCP/IP, multivendor networking, local or distributed name service, distributed network management

DECnet/OSI for OpenVMS provides the integration of DECnet and OSI network protocols that continues support for DECnet applications and enables support for OSI applications on OpenVMS. With a separate TCP/IP running on the same system, DECnet/OSI supports a multivendor, multiprotocol network environment. DECnet applications can be run over NSP, CLNP, or TCP/IP transports. OSI applications can be run over CLNP or TCP/IP transports.

A full implementation of the Digital Network Architecture (DNA) Phase V for OpenVMS systems, the OSI component of the DECnet/OSI software is implemented and tested in accordance with the current U.S. and UK GOSIP requirements. GOSIP is the Government OSI Profile that defines OSI capabilities required by government procurement.

For more information about DECnet/OSI, see the DECnet/OSI for OpenVMS documentation.

9.4 Interoperability in a VMScluster System

VMScluster systems for OpenVMS Alpha offer all the software features of VAXcluster systems. All the related VAXcluster software products, including the Business Recovery Server (BRS), are available for OpenVMS Alpha nodes. BRS, formerly named the Multi-Datacenter Facility, was released in June 1994.

In addition, VMScluster systems running OpenVMS Alpha Version 6.2 support the Small Computer System Interface (SCSI) interconnect as a multihost shared storage interconnect, a feature not available on OpenVMS VAX systems.

For configurations with VAX and Alpha nodes running the same version or different versions, additional capabilities and restrictions exist, as described in this section.

9.4.1 Booting in a Mixed-Architecture VMScluster System

The capabilities and restrictions of booting in a mixed-architecture VMScluster system are shown in Table 9–8.

Table 9–8 Booting in a Mixed-Architecture VMScluster System

Function	Description
System disk	Separate system disks required for VAX and Alpha systems
Cross-architecture satellite booting (VAX boot node provides booting services to Alpha satellites and Alpha boot node provides booting services to VAX satellites)	Enabled for systems running Version 6.1 or later of OpenVMS and DECnet for OpenVMS (Phase IV). Satellite booting across architectures is under investigation for nodes running DECnet/OSI.

9.4.2 Upgrades in a Mixed-Architecture VMScluster System

Rolling upgrades in a mixed-architecture VMScluster system are performed in the same way that rolling upgrades in a single-architecture VMScluster or VAXcluster system are performed. Cross-architecture upgrading is not enabled.

9.4.3 Restrictions of Selected Features in Mixed-Version VMScluster Systems

New features or improvements to existing features are introduced with each new release. Even though a feature or improvement is available on both platforms, its use may be restricted in a mixed-version or mixed-architecture VMScluster, if it was not available in earlier versions.

9.4.3.1 Process Identifiers Limit

Starting with Version 6.1, OpenVMS Alpha and OpenVMS VAX support up to 1024 identifiers. Previous versions of OpenVMS Alpha and OpenVMS VAX support up to 256 identifiers. If any nodes in a VMScluster system are running versions prior to Version 6.1, the limit of 256 identifiers is in effect.

9.4.3.2 Virtual I/O Cache

Virtual I/O cache became available in OpenVMS AXP Version 1.5 running standalone and OpenVMS VAX Version 6.0. If any nodes in a mixed-version or mixed-architecture VMScluster system are running OpenVMS VAX Version 5.5–2 or OpenVMS AXP Version 1.5, the virtual I/O cache disables itself.

9.4.3.3 Remote Monitoring

Remote monitoring is a feature of the Monitor utility (MONITOR) that enables you to monitor any node in a VMScluster system. You can do this either by issuing the MONITOR CLUSTER command or by adding the /NODE qualifier to any interactive MONITOR request.

Remote monitoring is limited across nodes running different versions of OpenVMS. Table 9–9 shows which versions enable this feature and which do not.

In addition, a second difference exists when you are monitoring remote nodes in a VMScluster. The limit on the number of disks that can be monitored remotely on OpenVMS VAX Version 6.2 and OpenVMS Alpha Version 6.2 was raised from 799 to 909 for record output and from 799 to 1817 for display and summary outputs. If you are monitoring a remote node running OpenVMS Version 6.2 from a system running an earlier version of OpenVMS, its limit of 799 is in effect.

Table 9–9 Remote Monitoring Compatibility in a VMScluster

	OpenVMS VAX Version 6.*n*	OpenVMS Alpha Version 6.*n*	OpenVMS Alpha Version 1.5 & VAX Version 5.*n*
OpenVMS VAX Version 6.*n*	Yes	Yes	No
OpenVMS Alpha Version 6.*n*	Yes	Yes	No
OpenVMS Alpha Version 1.5 and VAX Version 5.*n*	No	No	Yes

If you attempt to monitor a remote node that is incompatible, the following message is displayed:

```
%MONITOR-E-SRVMISMATCH, MONITOR server on remote node is an incompatible
version
```

If you receive this error message, you can still obtain data about the remote node with MONITOR. You do this by recording the data on the remote node and then using the MONITOR playback feature to examine it on the local node.

For more information on MONITOR, see the *OpenVMS System Management Utilities Reference Manual*.

9.4.4 ANALYZE/ERROR and ANALYZE/IMAGE in a Mixed-Architecture VMScluster System

The ANALYZE/ERROR (ERF) utility is architecture specific. In other words, in a mixed-architecture VMScluster, a user on a VAX node cannot use ANALYZE /ERROR to analyze the ERRLOG.SYS of an Alpha node, nor can a user on an Alpha node use ANALYZE/ERROR to analyze the ERRLOG.SYS of a VAX node.

Support for using ANALYZE/ERROR across architectures is under investigation.

On Alpha, using ANALYZE/IMAGE, you can analyze both Alpha and VAX images. On VAX, you can only analyze VAX images. If you attempt to analyze an Alpha image from an OpenVMS VAX system, an unclear message is displayed.

A correction to the message is planned for a future release of OpenVMS VAX. A change to the OpenVMS VAX functionality is under investigation.

9.4.5 VMScluster Configuration Support

OpenVMS Alpha and OpenVMS VAX provide two levels of support for mixed-version and mixed-architecture VMScluster configurations: warranted support and migration support.

Warranted support means that Digital has fully qualified the two versions to coexist in a VMScluster, and will answer all problems identified by customers using these configurations.

Migration support is a superset of the rolling upgrade support provided in earlier releases of OpenVMS and is available for mixes that are not warranted. Migration support means that Digital has qualified the versions for use together in configurations that are migrating in a staged fashion to a newer version of OpenVMS VAX or to OpenVMS Alpha. Digital will answer problem reports submitted against these configurations. However, in exceptional cases, Digital may request that you move to a warranted configuration as part of answering the problem. Migration support will help customers move to warranted VMScluster version mixes with minimal impact on their cluster environments.

For the level of support provided for all possible version pairings, see the current *OpenVMS New Features Manual*.

Note that Digital does not support the use of more than two versions in a VMScluster at a time. In many cases, more than two versions will successfully operate, but Digital cannot commit to resolving problems experienced with such configurations.

10 Guidelines for Developing Applications for Mixed-Architecture VMScluster Systems

These guidelines are provided to facilitate the evaluation and modification, if necessary, of applications intended to run in a mixed-architecture VMScluster system.

A basic assumption about applications running in a mixed-architecture VMScluster system is that they should work just as they do in a single-architecture VMScluster or VAXcluster system. VMScluster components, such as the lock manager, RMS, and the connection manager, behave the same in a mixed-architecture VMScluster as they do in a single-architecture VMScluster or VAXcluster system.

Most OpenVMS VAX applications require some migration work to run on an Alpha computer. The exceptions are the few applications that utilize only simple DCL commands, that is, they do not run any application images. In general, you need to at least recompile and relink your application so it will run on an Alpha computer.

For an application to work in a mixed-architecture VMScluster system, consider the following aspects of an application:

- User interface
- System management interface
- Interactions between the application executing on VAX and Alpha nodes
  - File format compatibility
  - Data packing
  - Data-type selection
  - Buffer size
  - Data access and locking
- Missing features

Neither users nor system managers should notice a difference in the behavior of the application, regardless of whether the node in a VMScluster system is a VAX or an Alpha computer. Users should expect to see better application performance on OpenVMS Alpha systems. System managers should expect to maintain architecture-specific copies of image files. Other differences should be minimal.

Application compatibility with mixed-architecture VMScluster systems can vary, depending on the degree of differences visible to users and system managers. The following sections should help you determine how well your application will function in a mixed-architecture VMScluster environment and what modifications may be necessary.

10.1 User Interface

User interfaces for a given application should be identical on both VAX and Alpha nodes. At a minimum, one implementation of the user interface should be a subset of the other. In addition, the differences may need to be noted in user documentation.

10.2 System Management

The system management aspects of an application pertain to the similarity of the installation and licensing mechanisms on both VAX and Alpha. The goal for the application developer is to minimize the work required for managing different architectural versions of applications that are running in the mixed-architecture VMScluster system.

10.3 File Format Compatibility

It is important to clearly differentiate between two kinds of files—image files (including shareable images (.EXE)) and application data files. Image files cannot be activated across architectures. An image built for a VAX computer will run only on a VAX computer and an image built for an Alpha computer will run only on an Alpha computer. If an application kit is to be shared by Alpha and VAX systems, it must either contain separate images for each platform, perform the link during the installation, or use DECmigrate for OpenVMS Alpha Systems to translate the application. Note that DECmigrate is an optional product, released separately from the OpenVMS VAX operating system.

A common kit will probably need conditional statements for linking, as the process for linking shareable images differs between VAX and Alpha systems. F$GETSYI ("ARCH_TYPE") can be used from DCL, as described in *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

Applications should be able to share data files across the Alpha and VAX systems; doing so is usually required for the application to run properly in a mixed-architecture VMScluster environment.

Note

If full data file compatibility cannot be achieved, applications should ensure that the incompatible format is recognized by both types of systems and an error message is issued.

10.4 Data Packing

Aligned data can provide significant performance improvements on VAX systems (up to 4 times faster) and on Alpha systems (up to 100 times faster). Unaligned data was typical on older VAX systems where minimal consumption of memory was very important and data was packed tightly together, ignoring alignment issues.

If you share data between Alpha and VAX systems, avoid unaligned data structures. Use at least longword alignment for in-memory data structures whenever possible; quadword alignment is preferred, if possible.

If you have a real-time application or an application that exhibits poor file I/O characteristics and shares data through a file, you may find it advantageous to align the file data structures.

For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

10.5 Data-Type Selection

Check to see if your application depends on the size or bit representation of an underlying data type. Ideally, all data accessed by different nodes in the VMScluster system should have the identical format on VAX and Alpha nodes. Examples include the following:

- Only ASCII, integer, or F_ and G_floating-point data formats are used in files (no H_floating, no IEEE floating, no machine instructions).

  For FORTRAN, this can be relaxed, because FORTRAN provides run-time support for automatic conversion between floating-point types via the /CONVERT compiler qualifier, and so forth.
- All structures stored in files are equally aligned for VAX and Alpha systems.

You can use different data formats if format translation is transparent to the user.

Support for floating-point data types differs between OpenVMS VAX and OpenVMS Alpha systems. For more information, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

You may want to use the IEEE floating-point formats on Alpha systems for coexistence with other open systems. However, using the IEEE floating-point formats may impede mixed-architecture VMScluster interoperability, because the T, S, and X IEEE floating-point data types are not supported on VAX systems.

For Fortran programs, this is not a major obstacle because of the /CONVERT compiler qualifier or the CONVERT= option on OPEN statements. For C programs, the automatic conversion of binary floating-point data in files is not possible, so this is an issue.

If your VAX application requires full 56-bit precision, that is, the D56_floating data type, or if it requires the H_floating data type, you can translate your image with DECmigrate. Because a translated image runs significantly slower than a native Alpha image, isolate the set of routines requiring 56-bit D_floating precision or the H_floating data type into its own small image for translation, and run the rest of the application native.

If your application contains data types that are not supported on OpenVMS Alpha, you can make the changes that are appropriate for it (either translation or recoding). You can then port your application to OpenVMS Alpha systems and share data between Alpha and VAX systems in a mixed-architecture VMScluster system. For more information about translation support, see the manual *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*. For more information about supported data types, see the documentation for the compiler you are using.

10.6 Buffer Size

If you need to make changes to your code, such as aligning data structures and using different data types, you need to review the buffer sizes in your application. Such changes generally require larger buffers.

10.7 Data Access and Locking

Data access is synchronized using the same resource names and locking sequences on VAX and Alpha nodes. This is not new for mixed-architecture VMScluster systems, because the need to ensure resource naming conventions is standard for VMScluster systems today. However, you should test your application to ensure correct operation.

10.8 Missing Features

Check your application for dependencies on features that might not exist on the target system (see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*). Also, check your application for dependencies on features provided by middleware applications that might not exist on the target system. Because a feature or application may not be available on both platforms, using the feature or application in a mixed-architecture VMScluster system could yield unanticipated results.

10.9 Application Compatibility Checklist

Use the following checklist to determine how well your application will work in a mixed-architecture VMScluster environment.

1. **User Interface**

   - ☐ Does the application present the same user interface on both architectures?

     If not, is one interface an easily described subset of the other?
   - ☐ Does the application use the same documentation on both architectures?
   - ☐ Are user interface differences documented for the user?
   - ☐ Are error messages the same across the architectures for the same error condition?

2. **System Management**

   - ☐ Are version numbers for feature-compatible versions identical for both architectures?
   - ☐ Is a single version of the License Management Facility (LMF) possible? Can both the VAX and Alpha versions of the application run using comparable licensing mechanisms? There should not be a requirement that the application use LMF Version 1.*x* on a VAX system and a different LMF version on an Alpha system.
   - ☐ Are separate installations required for each architecture?
   - ☐ Does the application use the same installation procedure and documentation?
   - ☐ Does the application work across architectures without additional system management steps?

- ☐ Does the application present comparable failure scenarios and recovery capabilities on both architectures?

3. **File Format Compatibility**

   - ☐ Are all files associated with the application accessible and used equally on each architecture? Remember that object and image libraries and image files are different between VAX and Alpha systems. (Other library types, such as .TLB and .HLB, do not present a problem. They are the same.)
   - ☐ Can files that do not contain machine instructions be used equally on either architecture?
     - Is full compatibility the default or is user action required?
     - How does the application manage differences in file formats?
     - If your application does not allow files to be shared, does it recognize an incompatible file format and present a suitable warning message?

4. **Data Packing**

   - ☐ Does your application use unaligned data structures?

     If so, what is the performance impact for your application?

5. **Data-Type Selection**

   - ☐ Does your product use D56_floating or H_floating on VAX?
     - If so, what is the performance impact if you use the emulation routines that provide full D56_floating or full H_floating support?
     - What is the precision-loss impact if you replace D56_floating with D53_floating?

6. **Buffer Size**

   - ☐ Will you need to change the alignment of any data structures or any data types in your application?

     If so, what are the implications on buffer size?

7. **Data Access and Locking Compatibility**

   - ☐ Does your product provide compatibility of resource names?
   - ☐ Does your product provide compatibility of locking sequences?

8. **Missing Features**

   - ☐ Does your product utilize features that are only available on one platform?
   - ☐ Are you documenting these differences to your users so they will know what will work in a mixed-architecture VMScluster?

Index