HP OpenVMS Version 8.3 New Features and Documentation Overview

Order Number: BA322-90046

July 2006

This manual describes the new features associated with the OpenVMS Alpha and OpenVMS for Integrity servers Version 8.3 operating systems and provides an overview of the documentation that supports this software.

Revision/Update Information:	This is a new manual.
Software Version:	OpenVMS I64 Version 8.3 OpenVMS Alpha Version 8.3

Hewlett-Packard Company
Palo Alto, California

ZK6679

The HP OpenVMS documentation set is available on CD.

This document was prepared using DECdocument, Version 3.3-1b.

Contents

Preface

Intended Audience

This manual is intended for general users, system managers, and programmers who use the HP OpenVMS operating system.

This document describes the new features related to Version 8.3 of the OpenVMS operating system. For information about how some of the new features might affect your system, read the release notes before you install, upgrade, or use Version 8.3.

Document Structure

This manual contains the following parts and chapters:

- Part I, OpenVMS Version 8.3 New Features
  - Chapter 1 summarizes the new OpenVMS software features.
  - Chapter 2 describes new features of interest to general users of the OpenVMS operating system.
  - Chapter 3 describes new features that are applicable to the tasks performed by system managers.
  - Chapter 4 introduces the COPY/RECORDABLE_MEDIA (CDDVD) utility.
  - Chapter 5 describes new features that support programming tasks.
  - Chapter 6 describes the InfoServer utility, which is now supported on OpenVMS Alpha as well as OpenVMS for Integrity servers.
  - Chapter 7 describes significant layered product new features.
- Part II, OpenVMS Documentation
  - Chapter 8 describes the OpenVMS documentation changes from the previous version.
  - Chapter 9 describes how the documentation is delivered.
  - Chapter 10 describes each manual in the OpenVMS documentation set.

Related Documents

For additional information about HP OpenVMS products and services, visit the following World Wide Web address:

`http://www.hp.com/go/openvms`

Reader's Comments

HP welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@hp.com**
Postal Mail	Hewlett-Packard Company OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How to Order Additional Documentation

For information about how to order additional documentation, visit the following Web site:

`http://www.hp.com/go/openvms/doc/order`

Conventions

The following conventions may be used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
$\boxed{\text{Return}}$	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.) In the HTML version of this document, this convention appears as brackets, rather than a box.
. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.

|	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold type	Bold type represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic type	Italic type 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 *dd* represents the predefined code for the device type).
UPPERCASE TYPE	Uppercase type indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Example`	This typeface indicates code examples, command examples, and interactive screen displays. In text, this type also identifies URLs, UNIX commands and pathnames, PC-based commands and folders, and certain elements of the C programming language.
-	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.

Part I

OpenVMS Version 8.3 New Features

1 Summary of HP OpenVMS Version 8.3 New Features

OpenVMS Version 8.3 delivers the highest possible levels of availability, scalability, flexibility, performance, and security that are required for operating in a 24x365 environment. OpenVMS continues to enhance its availability and performance by including new technology in the base operating system and in the OpenVMS Cluster software environment, as well as support for new HP Integrity servers.

1.1 Summary Table

OpenVMS Version 8.3 includes all the capabilities of OpenVMS Version 8.2 and Version 8.2–1 as well as the new features added to the OpenVMS operating system. Table 1-1 summarizes each feature provided by OpenVMS Version 8.3 and presents these features according to their functional component (general user, system management, programming, and associated products).

Table 1–1 Summary of OpenVMS Version 8.3 Software Features

General User Features

Features	Description
New supported Integrity server systems	Support for new entry-level, mid-range, and high-end Integrity servers.
Batch queue maximum increased	Job limit increased to 65535.
Ctrl/T support for remote process	Ability to define new symbol DCL$CTRLT_PID to point to a remote process ID.
Customized output of Ctrl/T	Ability to define new symbol DCL$CTRLT.
DCL commands and lexical functions	New and changed DCL commands, qualifiers, and lexical functions.
DCL permanent symbols	Two new symbols added to $FACILITY and $IDENT.
DCL prompt size increased	Maximum prompt size increased from 32 to 64 characters.
Keyword added to /SINCE qualifier	JOB_LOGIN keyword added.
Hyper-Threading (I64 only)	Allows processors to create a second virtual core that allows additional efficiencies of processing.
HP Instant Capacity (iCAP) and HP Temporary Instant Capacity (TiCAP) (I64 only)	New HP pricing solutions products for cell-based Integrity servers.
I/O size limit increased	Maximum block size limit increased for COPY command.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS Version 8.3 Software Features

General User Features

Features	Description
LMF enhancement	Ability to change location of target compliance report.
HP nPartition Provider (I64 only)	Now ported to OpenVMS Version 8.3
HP Pay per use (PPU) (I64 only)	HP On Demand Solutions product for cell-based Integrity servers.
HP Superdome servers support (I64 only)	Based on HP sx1000 chipset, OpenVMS can support configurations with both PA-RISC nPartitions and Intel® Itanium® 2 nPartitions in the same server.
Web-Based Enterprise Management Services for OpenVMS (WBEM)	Industry-standard enterprise management framework.

System Management Features

Features	Description
BACKUP Ctrl/T message	Expanded to include more information during an interactive backup operation
BACKUP /IO_LOAD qualifier	Provides the ability to increase or decrease the number of simultaneous read I/Os on your system
BACKUP /PROGRESS_REPORT qualifier	Sends backup information to the current output device
BACKUP save-set encryption	With the addition of the ENCRYPT and DECRYPT commands, uses an AES encryption algorithm and a user-specified key to encrypt and decrypt a BACKUP save set.
CD and DVD optical media tools	New tools for recording CD and DVD optical media.
Cluster interconnect	Provides data compression and multi-gigabit line speed.
Cluster satellite boot	Cluster satellite boot support is now available in OpenVMS for Integrity servers, as well as OpenVMS Alpha.
Dynamic lock remastering	The method by which decisions are made to remaster lock trees has been updated.
Dynamic volume expansion (DVE) support	Added to the BACKUP utility.
Encryption	The Encryption for OpenVMS layered product functionality has been integrated into the operating system.
InfoServer utility	Now supported on OpenVMS Alpha as well as OpenVMS for Integrity servers. Provides features of the InfoServer hardware product on AlphaServers and Integrity servers running OpenVMS. The InfoServer utility can be used to upgrade or install the OpenVMS operating system on AlphaServers or Integrity servers.
MONITOR ALIGN command (I64 only)	New command added to the Monitor utility.
MONITOR class name qualifiers for PROCESSES class	Used to monitor per-process-based modes usage.
Multipath enhancement	New active-active feature of EVA and MSA controllers added.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS Version 8.3 Software Features

System Management Features

Features	Description
Patch-related menu option	OpenVMS operating system distribution media main menu now includes a new option to perform patch-related operations.
PCSI utility enhancements	New features include manual verification, automatic verification, full ODS-5 support, and validation of signed product kits.
SANCP utility	Allows you to limit the number of active I/Os a host may have across all paths to logical unit numbers (LUN) on a given Fibre Channel storage port.
SAS utility (I64 only)	Configures Integrated RAID (IR) functionality for the HP 8 Internal Port Serial Attached SCSI Host Bus Adapter (SAS Controller)
SCACP data compression	SET VC/COMPRESSION (or /NOCOMPRESSION) commands enable or disable sending compressed data by specified virtual circuits (VCs).
SCACP multi-gigabit scaling	Allows use of /WINDOW=RECEIVE_SIZE and /WINDOW=TRANSMIT_SIZE qualifiers to override the automatically calculated receive and transmit window sizes.
HP OpenVMS I64 Serial Multiplexer (MUX) Support for Integrity servers (I64 only)	Allows use of USB to add serial lines to Integrity servers.
Spinlock Trace Utility	Display enhanced.
HP System Analysis Tools utilities	System Dump Debugger is now available on OpenVMS for Integrity servers. Several new System Dump Analyzer commands and qualifiers have been added.
System parameters	Several new system parameters have been added to this release: • EXECSTACKPAGES • GB_CACHEALLMAX • GB_DEFPERCENT • IO_PRCPU_BITMAP • LOCKRMWT • SCH_HARD_OFFLD • SCH_SOFT_OFFLD • SCHED_FLAGS • SMP_CPU_BITMAP • SMP_PRCPU_BITMAP • VCC_PAGESIZE • VCC_RSVD

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS Version 8.3 Software Features

System Management Features

Features	Description
System Service Logging	SSLOG has been enhanced.
Time zones	Additional time zones now supported by adding them to the database.
UCM changes for device management	Automatically configures devices that you plug into the system.
Virtual LAN (VLAN) support in OpenVMS	Adds support for IEEE 802.1Q (VLAN) in OpenVMS.
HP Volume Shadowing for OpenVMS enhancements	Addition of these new features: • Automatic bitmap creation on volume processing • New SET SHADOW qualifier /RESET

Programming Features

Features	Description
Common Data Security Architecture (CDSA)	New features include Secure Delivery and support for Human Recognition Service Standard (HRS).
HP C Run-Time Library (CRTL) enhancements	The following enhancements have been added: • Symbolic link and POSIX-compliant pathnames • Byte-range locking • New functions • TCP/IP header file updates
Deadlock wait	New item code, PPROP$C_DEADLOCK_WAIT, provides the ability to use sub-second deadlock wait for the lock manager.
Items codes	Addition of these new item codes: • DEVICE_MAX_IO_SIZE • VOLUME_RETAIN_MAX • VOLUME_RETAIN_MIN • MAILBOX_INITIAL_QUOTE • MAILBOX_BUFFER_QUOTA
HP Kerberos for OpenVMS enhancements	New features added to Kerberos for OpenVMS. Kerberos Version 3.0 for OpenVMS is based on MIT Kerberos V5 Release 1.4.1.
Library utility enhancement (I64 only)	Ability to list demangled and mangled names.
Linker utility enhancement	/DNI qualifier and DEMANGLED_SYMBOLS keyword for /FULL qualifier.
HP MACRO Compiler for OpenVMS Alpha systems enhancements	Upgraded to use the latest GEM backend for Alpha systems, added a new /ARCHITECTURE DCL qualifier, and other enhancements.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS Version 8.3 Software Features

Programming Features

Features	Description
Record Management System (RMS) enhancements	Several enhancements made to RMS.
HP Secure Sockets Layer (SSL) for OpenVMS enhancements	New features added to SSL. HP SSL Version 1.3 is based on OpenSSL 0.9.7e
USB generic driver	Provides a $QIO-based API for users to add support for USB devices.

Associated Products Features

Features	Description
HP Distributed NetBeans for OpenVMS	Allows running the NetBeans IDE on desktop system.
HP OpenVMS Management Station Version 3.3	OpenVMS Management Station Version 3.3 is included with OpenVMS Alpha Version 8.3.
HP Secure Web Browser for OpenVMS	Version 1.7-13 provided with OpenVMS Version 8.3.
HP Secure Web Server for OpenVMS	Version 2.1 provided with OpenVMS Version 8.3.
HP TCP/IP Services for OpenVMS	Version 5.6 supported on OpenVMS Version 8.3.
HP Web Services Integration Toolkit for OpenVMS	Provides a set of individual tools.

Prior to getting started with OpenVMS Version 8.3, see the *HP OpenVMS Version 8.3 Release Notes* and the *HP OpenVMS Version 8.3 Upgrade and Installation Manual*.

2
General User Features

This chapter provides information about new features for all users of the HP OpenVMS Alpha and OpenVMS for Integrity servers operating systems.

2.1 New Integrity Server Support

OpenVMS Version 8.3 supports new entry-level, midrange, and high-end Integrity server systems. In addition, OpenVMS Version 8.3 continues to support all of the Integrity servers and options supported in OpenVMS Version 8.2 and Version 8.2–1.

2.2 Batch Queue Job Limit Increased

In OpenVMS Version 8.3, the maximum job limit on batch queues is increased from 255 to 65535.

2.3 DCL Commands and Lexical Functions

Table 2–1 and Table 2–2 summarize new and changed DCL commands, qualifiers, and lexical functions for OpenVMS Version 8.3. A few additional new features related to DCL usage are described in the following sections. For more information, see the online help or the *HP OpenVMS DCL Dictionary*.

Table 2–1 Updates to DCL Commands and DCL Documentation

DCL Command	Documentation Update
DEASSIGN	New /[NO]LOG qualifier.
DECRYPT	New command.
DIFFERENCES	New WHITE_SPACE keyword for /IGNORE.
ENCRYPT	New command.
READ	New /WAIT qualifier; new LT and LE keywords for /MATCH.
SEARCH	New keywords for /WILDCARD qualifier.
SEARCH	/STATISTICS qualifier now defines several DCL symbols with the statistics information.
SET	New /RMS_RELATED_CONTEXT qualifier.
SET FILE	Seven new *DATE keywords added to the /ATTRIBUTES qualifier table. Five new global buffer options added for RMS.

(continued on next page)

Table 2–1 (Cont.) Updates to DCL Commands and DCL Documentation

DCL Command	Documentation Update
SHOW DEVICES	Support added to display the last path switch times for multipath devices when /FULL is specified. For LAN devices, the /FULL qualifier updated to display default and current MAC address information, the LAN protocol in use (if applicable), the speed of the data link, and various other enabled characteristics, such as auto-negotiation, duplex mode, and jumbo frames.
SHOW LICENSE	Command updated to show all licenses in the OE database to be displayed with one command.
SHOW PROCESS	Q key option added to /CONTINUOUS qualifier.
SYNCHRONIZE/TIME_ OUT=n	New command that allows the user to specify the number of seconds to wait before terminating the SYNCH command.

Table 2–2 Updates to DCL Lexicals and Lexicals Documentation

DCL Lexical	Documentation Update
F$CUNITS	New lexical function.
F$FILE_ATTRIBUTES	Two new item codes, GBC32 and GBCFLAGS.
F$GETDVI	New item codes added.
F$LICENSE	New optional argument to specify license producer.
F$MATCH_WILD	New lexical function.

2.3.1 Ctrl/T Support for Remote Process

You can define the new symbol DCL$CTRLT_PID to point to a remote process ID. If you have the necessary privileges, you can then display Ctrl/T information for that process. The remote process can be a different process on the same system or a process on a different system in the cluster. The following example shows how a privileged user on NODE1 can define DCL$CTRLT_PID to view process information for user JSMITH on NODE2.

```
$ <Ctrl/T>
NODE1::SYSTEM 17:40:55   (DCL)   CPU=00:00:00.16 PF=212 IO=98 MEM=146
$
$ DCL$CTRLT_PID="23800436"  !Define symbol to point to remote process ID
$
$ <Ctrl/T>
NODE2::JSMITH 17:41:12 LOOPER    CPU=01:28:05.17 PF=2700 IO=594 MEM=322
$
```

2.3.2 DCL Permanent Symbols

On an image rundown, DCL populates the $SEVERITY and $STATUS symbols. With Version 8.3, two new symbols are added, $FACILITY and $IDENT, which contain the facility number and the message number.

```
$ EXIT  %X10911A02
$ SHOW SYMBOL $STATUS
  $STATUS == "%X10911A02"
$ SHOW SYMBOL $FACILITY
  $FACILITY == "%X00000091"
$ SHOW SYMBOL $IDENT
  $IDENT == "%X00000340"
$ SHOW SYMBOL $SEVERITY
  $SEVERITY == "2"
```

2.3.3 Customizing the Output of Ctrl/T

By defining the new symbol DCL$CTRLT, users can augment the traditional Ctrl/T output with the text defined for DCL$CTRLT. This technique can be used to indicate progress in user applications or for debugging purposes. The following example demonstrates the use of DCL$CTRLT within a command procedure. The command procedure is running in a loop that updates the symbol DCL$CTRLT to indicate the number of loop iterations executed so far.

```
$ TYPE CTRLT_LOOP.COM
$ inner=0
$ outer=0
$ loop:
$ loop1:
$ if inner .gt. 20000 then goto end_loop1
$ inner=inner+1
$ dcl$ctrlt=F$FAO("Inner loop count is !SL !/ -
_$ Outer loop count is !SL",inner,outer)
$ goto loop1
$ end_loop1:
$ inner=0
$ outer=outer+1
$ goto loop
$
$ @CTRLT_LOOP
```

```
NODE1::JSMITH 10:46:37   (DCL)   CPU=00:03:42.68 PF=13453 IO=6743 MEM=187
Inner loop count is 12306
 Outer loop count is 0
NODE1::JSMITH 10:46:43   (DCL)   CPU=00:03:49.19 PF=13455 IO=6744 MEM=187
Inner loop count is 19200
 Outer loop count is 2
  .
  .
  .
```

2.3.4 JOB_LOGIN Keyword Added to /SINCE Qualifier

You can now specify /SINCE=JOB_LOGIN in any command that accepts the /SINCE qualifier (except SHOW LICENSE). JOB_LOGIN refers to the login time of the master process in the job. For example, PIPE creates a subprocess for each pipe segment; therefore /SINCE=LOGIN cannot be effectively used in a pipe.

```
$ PIPE DIRECTORY/SINCE=LOGIN | SEARCH SYS$INPUT TEST
%SEARCH-I-NOMATCHES, no strings matched
$
$ PIPE DIRECTORY/SINCE=JOB_LOGIN | SEARCH SYS$INPUT TEST
TEST.TXT;1
```

2.3.5 I/O Size Limit Increased for COPY Command

The maximum per-I/O block count supported for the COPY command has been increased to 2**31 –1. Note that the block count is reduced if necessary to the maximum supported by the device driver performing the I/O. For example, the maximum I/O size supported by SCSI Fibre Channel drivers is currently 256 blocks.

2.3.6 Increase Maximum Prompt Size

The maximum DCL prompt size has been increased from 32 characters to 64 characters to accommodate special prompts and escape sequences, which some users require.

2.4 Hyper-Threading (I64 Only)

OpenVMS for Integrity servers Version 8.3 supports Hyper-Threading with nPartitions on dual-core Intel Itanium 2 processors. Hyper-Threading provides the ability for processors to create a second logical CPU on a single core that may allow additional efficiencies of processing. For example, a dual-core processor with Hyper-Threading active provides four logical CPUs, two on each core.

The EFI Shell `cpuconfig` command can enable and disable Hyper-Threading for an nPartition whose processors support it. Recent releases of the nPartition Command and Partition manager also support Hyper-Threading.

The effect that hyperthreads have on performance depends heavily on the application mix that is running. HP recommends that you start with hyperthreads turned off and experiment later. Two CPUs that share a core when hyperthreading is enabled are referred to as **cothreads**. The SHOW CPU/BRIEF and SHOW CPU/FULL commands now provide information about cothreads. For example:

```
$ show cpu/brief 3

System: XYZZY, HP rx4640

CPU 3    State: RUN              CPUDB: 820DB480     Handle: 000060A0
         Owner: 000004CB8      Current: 000004C8     Partition 0 (XYZZY)
         COTHd:          1
```

In this example, `COTHd: 1` indicates that CPU 3 and CPU 1 share the same core.

A sample `cpuconfig` command output looks like this:

```
Shell> cpuconfig

PROCESSOR MODULE INFORMATION

        # of              L3      L4      Family/
CPU     Logical           Cache   Cache     Model          Processor
Module     CPUs   Speed   Size    Size     (hex.)    Rev   State
-----   -------   ------  ------  ------  ---------  ---   -----
 0           4    1.4 GHz  6 MB   None    20/00      CO    Active

CPU threads are turned on.
```

For information about the `cpuconfig` command, see the *HP OpenVMS Version 8.3 Upgrade and Installation Manual*. For information about Hyper-Threading and nPartitions, see the *HP System Partition Guide: Administration for nPartitions*.

2.5 HP Instant Capacity (iCAP) and HP Temporary Instant Capacity (TiCAP) (I64 Only)

OpenVMS Version 8.3 now supports iCAP, which is the HP Utility Pricing Solutions product for cell-based Integrity servers that has a pricing model based on purchasing components (processors, cell boards, and memory). With Instant Capacity you initially purchase a specified number of activated components and pay a Component without Usage Rights (CWUR) fee for a specified number of deactivated components. To activate a component, you pay the balance of the component price and license it through the application of a codeword obtained from the secure iCAP web portal. Components can be instantly activated without requiring a reboot.

OpenVMS Version 8.3 also supports TiCAP, an HP product that enables customers to purchase prepaid processor-activation rights for a specified (temporary) period of time. Temporary capacity is in increments, such as 20-day or 30-day increments, where a day equals 24 hours for a core.

For more information about iCAP and TiCAP, see the *HP Instant Capacity User's Guide*, at the following Web site:

`http://docs.hp.com/en/hplex.html#Utility%20Pricing`

2.6 License Management Facility (LMF) Changes and Enhancements

The following sections describe the changes and enhancements to the License Management Facility.

2.6.1 LMF Compliance Report

You can now change the target account for LMF compliance reports. LMF sends the compliance report to the account defined by the logical name LMF$COMPLIANCE_CONTACT_ACCOUNT, but the default account is still the SYSTEM account.

2.6.2 License Terminology Change (I64 Only)

The licensing and terminology for OpenVMS for Integrity servers has changed from Per Processor License (PPL) to Per Core License (PCL). With the introduction of the dual-core Intel Itanium 2 processor, the term "processor" does not mean what it meant previously. With dual-core Intel Itanium 2 systems, each processor contains 2 cores, and these systems are licensed by the number of active cores. On a system other than a dual-core Intel Itanium 2 system, core equals processor.

PCL implements the licensing model on OpenVMS for Integrity servers. In the PCL model, a product is licensed according to the number of active processor cores on the system, not the static rating scheme used for Alpha and VAX systems. Each active processor core requires one PCL unit. If you increase or decrease the number of active processor cores on a system, the requirement for PCL licenses changes.

A PCL license is required to run operating environments, OE products purchased separately (like clustering), and many standalone products on OpenVMS for Integrity servers.

PCL licenses offer flexibility because you can purchase licenses in the exact number you need, and you can move the licenses to other processors. If you upgrade or reconfigure your system with additional processor cores, you purchase additional PCL licenses.

LMF constantly checks the number of PCL licenses against the number of active processor cores and enforces a soft compliance model. Any changes to the system are noted and are checked for compliance.

To implement PCL, the Hardware_ID option CPU_SOCKETS=*n* is changed to SOCKETS=*n*. Also, the SHOW LICENSE/CHARGE command is updated to display the number of active cores on the system.

The *HP OpenVMS License Management Utility Manual* has not been updated for this release with the new terminology. As you read the manual, take note of the following terminology changes:

- Per Processor License is now Per Core License.
- PPL is now PCL.
- CPU is now processor core.
- CPU_SOCKETS=*n* is now SOCKETS=*n*.

Note the following definitions:

- Processor — The component that plugs into a processor socket. The processor can contain more than one processor core.
- Processor module — The packaging of one or more processors to connect into a single socket on a system bus.
- Core — The actual data-processing engine within a cell-based processor. A single processor can contain multiple cores.
- Processor socket — The system board socket into which a processor attaches.

Note

Any previous PPL licenses continue to be supported, and no changes are required. You can also combine PCL licenses with the existing PPL licenses.

2.7 HP nPartition Provider for OpenVMS (I64 Only)

The HP nPartition Provider is now ported to OpenVMS Version 8.3 in support of Instant Capacity (iCAP) features on cell-based Integrity servers.

Note, the OpenVMS Version 8.3 nPartition Provider does not support local or remote nPartition management tasks, such as remote WBEM connections to the nPartition Provider running on OpenVMS Version 8.3.

2.8 HP Pay per Use (PPU) (I64 only)

OpenVMS Version 8.3 supports Pay per use for cell-based Integrity server systems leased through HP Finance. PPU enables customers to pay for only the processing cycles that they actually consume. There are two types of PPU:

- Percent CPU

  Continuously monitors the utilization of every CPU in the system.
- Active CPU

  Continuously counts the number of active CPUs in the system.

The system manager can instantly activate inactive CPUs to cope with additional loads. With both Percent CPU and Active CPU, the utilization data is sent to a separate Utility Meter and then to the secure HP PPU Web portal, where you can obtain various usage reports within 48 hours. (Note that Instant Capacity and Pay per use are mutually exclusive on any cell-based Integrity server.)

For more information about PPU, see the *HP Pay per use (PPU) User's Guide*, at the following Web site:

`http://docs.hp.com/en/hplex.html#Utility%20Pricing`

2.9 HP Superdome Hybrid Servers Support (I64 Only)

HP Superdome servers based on the HP sx1000 chipset can support configurations with both PA-RISC nPartitions and Intel Itanium 2 nPartitions in the same server.

Specific hardware, firmware, operating systems, and management tools are required for including both PA-RISC nPartitions and Intel Itanium 2 nPartitions on Superdome hybrid servers.

The HP OpenVMS for Integrity servers Version 8.3 release is supported on Superdome hybrid servers in an nPartition with the Intel Itanium 2 single-core processor with 9 MB cache.

For details and requirements, see the *HP Superdome Hybrid Servers: Intel Itanium 2 and PA-RISC nPartition Mixing* on the following Web site:

`http://docs.hp.com`

This document is available in the Systems Hardware area, under the HP Integrity Superdome Server and HP 9000 Superdome Server headings.

2.10 HP Web-Based Enterprise Management Services for OpenVMS (WBEM)

WBEM is an optional product available on HP OpenVMS for Integrity server systems that provides an industry-standard enterprise management framework and resource description. The WBEM structured framework is capable of being extended and uses Internet standards. Developers of management applications can take advantage of work previously done to expose resource information and operations. For example, developers can take code that was developed for a specific platform or application and use it with WBEM for the same purposes.

3 System Management Features

This chapter provides information about new features, changes, and enhancements for system managers.

3.1 BACKUP Utility Enhancements

OpenVMS Version 8.3 contains several enhancements to the Backup utility:

- Dynamic volume expansion (DVE)
- Save-set encryption
- A more complete Ctrl/T message displayed during an interactive backup operation
- A new /PROGRESS_REPORT qualifier that sends the expanded BACKUP messages to the current output device
- A new /IO_LOAD qualifier that affects the number of simultaneous READ I/Os on your system

For more information about all of these enhancements, see the *HP OpenVMS System Management Utilities Reference Manual*.

3.1.1 Dynamic Volume Expansion (DVE) Support in OpenVMS Backup Utility

Beginning with OpenVMS Version 8.3, volume expansion size is recorded in the save-set header when you perform image backup to a save set. Previously, BACKUP had no knowledge of DVE when initializing an output volume. This meant that, in performing disk-to-disk backup or restoring a save set, BACKUP did not preserve the volume expansion size or the logical volume size.

3.1.1.1 Volume Expansion Size

BACKUP/LIST now displays the volume expansion size if it exists in the save set. When you restore a save set (or perform a disk-to-disk backup), the target device inherits the volume expansion limit from the save set. If the save set does not include expansion size, BITMAP.SYS is sized as it was in previous versions of OpenVMS.

/IGNORE=LIMIT Option

A new option, /IGNORE=LIMIT, prevents the target device from inheriting the expansion limit.

/LIMIT Qualifier

The new qualifier /LIMIT allows you to specify the expansion size limit during restore or save operations regardless of the value stored in the save-set header, which matches how the /LIMIT qualifier of the INITIALIZE utility works.

3.1.1.2 Logical Volume Size

By default, the logical volume size is not preserved because restoring a save set of 2GB to a 4GB disk, for example, would result in only 2GB of available disk space.

/SIZE Qualifier

To preserve the logical volume size, use the /SIZE qualifier. When you specify /SIZE, the geometry of the target device is determined by the logical size rather than calling $GETDVI to find out the physical limits of the device.

The /SIZE qualifier uses an optional value as the new logical size of the target device. This new value overrides any existing value in the save set, which matches how the /SIZE qualifier in the INITIALIZE utility works.

/NOINITIALIZE Qualifier

A restriction in using BACKUP/NOINITIALIZE is that this command does not preserve the DVE characteristics of the output device. The reason is that the target device is mounted foreign, preventing OpenVMS from obtaining the expansion size and the logical size. To overcome this restriction, use the /LIMIT and /SIZE qualifiers.

The chapter “Managing Storage Media” in *HP OpenVMS System Manager’s Manual* contains a section that explains DVE in more detail.

3.1.2 Encrypting BACKUP Save Sets

The OpenVMS Backup utility provides protection against file or volume corruption by creating functionally equivalent backup copies. BACKUP creates save sets, which are written in BACKUP format so that only BACKUP can interpret the data in a save set. When you create save sets, you can protect them further by encrypting them.

OpenVMS Version 8.3 includes the following new encryption features:

- Support for Advanced Encryption Standard (AES) encryption algorithms
- Support for the following AES keywords for the ALGORITHM option:

  ```
  AESCBC          AESCFB
  AESECB          AESOFB
  ```

  Each of these modes can have a user-defined secret key in one of three different lengths (128, 192, and 256) for a total of 12 possible key combinations. If you specify =AES after the ALGORITHM option, the default is AESCBC128.
- Ability to encrypt the data in a save set using the AES encryption algorithm specified by the user.

Note

Standalone BACKUP, which is a version of the Backup utility that runs without the support of the OpenVMS operating system, does not support the /ENCRYPT qualifier.

3.1.3 Additional CTRL/T Messages

When you use BACKUP to back up or restore data interactively, you can press Ctrl/T to display the progress of the operation. In OpenVMS Version 8.3, this information is increased in one of the following situations:

- When you restore a save set from disk
- When you perform an image backup (using save set) to tape or disk

The additional information now displayed is the following:

- The percentage of the operation already completed
- An estimate of the time required to complete the operation

You can use the new /PROGRESS_REPORT qualifier to send the expanded message to the current output device.

3.1.4 New /PROGRESS_REPORT Qualifier

When you include the new /PROGRESS_REPORT qualifier while performing the BACKUP operations described in Section 3.1.3, a message indicating the progress of the BACKUP operation is sent to the output device.

3.1.5 New /IO_LOAD Qualifier

In OpenVMS Version 8.3, BACKUP is optimized to work more efficiently with new storage controllers. You can use the /IO_LOAD qualifier to increase or decrease the number of simultaneous READ I/Os that occur on your system.

The format of the command and qualifier is the following:

```
BACKUP /IO_LOAD=n
```

where *n* is an integer whose value can be between 1 and the process AST limit. The default value is 8, which is used if the /IO_LOAD qualifier is omitted from the command line.

3.2 CD and DVD Optical-Media Recording Tools

OpenVMS Version 8.3 supports new tools for recording CD and DVD optical media. These tools permit OpenVMS users to easily and directly record locally mastered disk volumes or disk image files onto a CD-R, CD-RW, DVD+R, or DVD+RW optical-media recording device. The resulting optical-media data disks generated by the recording tools can be used as part of data-archiving operations, mastering software distribution, and similar tasks.

The COPY/RECORDABLE_MEDIA command and related tools and diagnostics are intended to supplement and to eventually replace existing uses of the SYS$MANAGER:CDRECORD.COM CD optical-media recording tool.

For the COPY/RECORDABLE_MEDIA command, see the *HP OpenVMS System Management Utilities Reference Manual* and for support requirements and capabilities, see Chapter 4 in this manual.

3.3 OpenVMS for Integrity Servers Cluster Satellite Support

OpenVMS Version 8.3 supports cluster satellite booting for OpenVMS for Integrity server (I64) systems. The manner and requirements for I64 satellite systems differs greatly from those for Alpha. Read this section thoroughly before attempting to add an I64 system to your cluster.

3.3.1 Differences between Alpha and I64 Satellites

Table 3–1 lists the differences between Alpha and Integrity server satellites.

Table 3–1 Differences Between Alpha and Integrity Server Satellites

	Alpha	Integrity Servers
Boot Protocol	MOP	PXE (BOOTP/DHCP/TFTP)
Crash Dumps	May crash to remote system disk or to local disk via Dump Off the System Disk (DOSD)	Requires DOSD. Crashing to the remote disk is not possible.
Error Log Buffers	Always written to the remote system disk.	Error log buffers are written to the same disk as DOSD.
File protections	No different than standard system disk.	Requires that all loadable execlets are W:RE (the default case) and that certain files have ACL access via the VMS$SATELLITE_ ACCESS identifier.

Satellite

Any OpenVMS Version 8.3 system or a nPartition of a cell-based system can be used as a satellite. Support for nPartitions may require a firmware upgrade.

Satellite boot is supported over the core I/O LAN adapters only. All satellite systems must contain at least one local disk to support crash dumps and saving of the error log buffers across reboots. Diskless systems will not be able to take crash dumps in the event of abnormal software termination.

Boot Server

All Integrity server systems supported by OpenVMS Version 8.3 are supported as boot servers. At this time, HP does not support cross-architecture booting for Integrity server satellite systems, so any cluster containing Integrity server satellite systems must have at least one Integrity server system to act as a boot node as well.

Required Software

- OpenVMS Version 8.3
- HP TCP/IP Services for OpenVMS Version 5.6 or later

As with other satellite systems, the system software is read off of a disk served by one or more nodes to the cluster. The satellite system disk may be the same as the boot server´s system disk but need not be. Unlike with Alpha satellites, where it was recommended but not required that the system disk be mounted on the boot server, I64 satellite systems require that the system disk be mounted on the boot server.

TCP/IP must be installed on the boot server´s system disk. OpenVMS Version 8.3 must be installed on both the boot server´s system disk and the satellite´s system disk if different.

TCP/IP must be configured with BOOTP, TFTP and one or more interfaces enabled. At least one configured interface must be connected to a segment visible to the satellite systems. The boot server and all satellite systems will require an IP address. Please see the *HP TCP/IP Services for OpenVMS Version 5.6 Installation and Configuration* for details about configuring TCP/IP Services for OpenVMS.

3.3.2 Collecting Information from the Satellite System

If the satellite has a local disk with a version of OpenVMS installed, log in. If not, you may boot the installation DVD and select option 8 (Execute DCL commands and procedures) and execute the following commands:

```
$ LANCP :== $LANCP
$ LANCP SHOW CONFIG
```

```
LAN Configuration:
Device Parent Medium/User Version Link Speed Duplex Size  MAC Address       Current Address   Type
------ ------ ----------- ------- ---- ----- ------ ----  ----------------  ----------------  ----
EIB0          Ethernet    X-16    Up   1000  Full   1500  00-13-21-5B-86-49 00-13-21-5B-86-49 UTP i82546
EIA0          Ethernet    X-16    Up   1000  Full   1500  00-13-21-5B-86-48 00-13-21-5B-86-48 UTP i82546
```

Record the MAC address for the adapter you will use for booting. You will need it when defining the satellite system to the boot server. If the current address differs from the MAC address, use the MAC address.

3.3.3 Setting up the Satellite System for Booting and Crashing

If the satellite has a local disk with a version of OpenVMS installed, log in. If not, you may boot the installation DVD and select option 8 (Execute DCL commands and procedures.) Use SYS$MANAGER:BOOT_OPTIONS.COM to add a boot menu option for the network adapter from which you are booting. The procedure will ask you if this network entry is for a satellite boot and if so, it will set the Memory Disk boot option flag (0x200000) for that boot menu entry. The memory disk flag is required for satellite boot.

If you intended to use the system primarily for satellite boot, place the network boot option at position 1. The satellite system also requires DOSD (Dump Off the System Disk) for crash dumps and saving the unwritten error log buffers across reboots and crashes. BOOT _OPTIONS.COM may also be used to manage the DOSD device list. You may wish to create the DOSD device list at this time. See the *HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems* for information about setting up a DOSD device list.

3.3.4 Defining the Satellite System to the Boot Server

I64 Satellite systems boot via the PXE protocol. On OpenVMS, PXE is handled by BOOTP from the TCPIP product. If you are using more than one I64 boot server in your cluster, be sure the BOOTP database is on a common disk. See the TCPIP documentation for information on configuring TCPIP components. TCPIP must be installed, configured and running before attempting to define a satellite system.

On an I64 boot server, log in to the system manager´s or other suitably privileged account. Execute the command procedure SYS$MANAGER:CLUSTER_CONFIG_ LAN.COM. (CLUSTER_CONFIG.COM, which configures satellite nodes using DECnet, does not support I64 systems. It will, however, automatically invoke

CLUSTER_CONFIG_LAN for I64 systems.) CLUSTER_CONFIG_LAN is a menu-driven command procedure designed to help you configure satellite systems. The menus are context-sensitive and may vary depending on architecture and installed products. If you are unfamiliar with the procedure, please see refer to the System Management documentation for a more extensive overview of CLUSTER_CONFIG_LAN.

The essential information required to add an I64 satellite includes the node´s SCS node name, SCS system ID, and hardware address. In addition, you will need to know the satellite´s IP address, network mask, and possibly gateway addresses. If you are unfamiliar with these concepts, please refer to the TCPIP documentation. The procedure will create a system root for the satellite.

CLUSTER_CONFIG_LAN should perform all steps required to make the satellite system bootable. If you choose local paging and swapping files, you will be prompted to boot the satellite system into the cluster so that the files may be created. If not, paging and swapping files will be created on the served system disk and you may boot the satellites at your convenience.

3.3.5 Booting the Satellite

If you have previously added an option to the boot menu, select that option. If you haven't, please see your hardware documentation for the steps required to boot from a network adapter. Be sure to set the environment variable VMS_FLAGS to include the memory disk boot flag (0x200000). The system will detail boot progress in the form of a system message when VMS_LOADER is obtained from the network, followed by one period character written to the console device for every file downloaded to start the boot sequence and last by a message indicating that IPB (the primary bootstrap image) has been loaded.

Note the following example:

```
Loading.: Satellite Boot EIA0 Mac(00-13-21-5b-86-48)
Running LoadFile()

CLIENT MAC ADDR: 00 13 21 5B 86 48
CLIENT IP: 16.116.43.79  MASK: 255.255.248.0  DHCP IP: 0.240.0.0

TSize.Running LoadFile()

Starting: Satellite Boot EIA0 Mac(00-13-21-5b-86-48)
Loading memory disk from IP 16.116.43.78
........................................................................
Loading file: $13$DKA0:[SYS10.SYSCOMMON.SYSEXE]IPB.EXE from IP 16.116.43.78
%IPB-I-SATSYSDIS, Satellite boot from system device $13$DKA0:

    HP OpenVMS Industry Standard 64 Operating System, Version V8.3
    © Copyright 1976-2006 Hewlett-Packard Development Company, L.P.
```

Upon first full boot, the satellite system will run AUTOGEN and reboot.

3.3.6 Additional Tasks on the Satellite System

If you had not done so previously, create the dump file for DOSD at this time. Edit the SYS$STARTUP:SYCONFIG.COM file and add commands to mount the DOSD device. In order for the error log buffers to be recovered, the DOSD device must be mounted in SYCONFIG.

3.4 Dynamic Lock Remastering—LOCKRMWT

The method by which OpenVMS makes decisions to remaster lock trees is updated for OpenVMS Version 8.3. Prior to Version 8.3, the decision-making method was based on the system parameter LOCKDIRWT. Lock trees migrated to nodes with higher LOCKDIRWT values. If nodes had the same value of LOCKDIRWT, then lock trees migrated to the node with higher activity—the amount of higher activity was an extremely small hard-coded threshold and often resulted in lock trees thrashing between nodes.

Version 8.3 implements a new system parameter, lock master weight (LOCKRMWT). This parameter has a range of zero (0) to 10 and a default value of 5. The value of this parameter indicates the weight of the node's willingness to master lock trees. The larger the weight, the greater the likelihood of a tree moving to the node. The values of 0 and 10 are special. A zero indicates that a node does not want to master trees unless the node is the only node with any locks on the resource tree. Any trees mastered on a node with a zero are moved to an interested node as long as the interested node has a LOCKRMWT greater than 0. The value of 10 indicates that a node always wants to master lock trees. If a node with a LOCKRMWT lower than 10 masters a lock tree and a node with a value of 10 is interested, the lock tree remasters to the node with a value of 10.

In other cases, the difference between the current master and the remote node's LOCKRMWT is computed. The larger the difference, the greater the likelihood of the tree being remastered. In the case of nodes having the same LOCKRMWT, the tree is remastered when there is approximately 13% more activity on the remote node. If the remote node had an 8 and the current master a 5, the difference is 3 in this case, the lock tree would move to the remote node at about 2% more activity. In the case of the remote node having a 1 and the current master a 9, then the difference is -8. In this particular case, the lock tree is remastered to the remote node even if doing about 200% more activity than the current master.

The new LOCKRMWT parameter is dynamic and thus can be changed with SYSGEN on the running system. In addition, lock remastering still honors the PE1 system parameter) where a lock tree with more locks than the value in PE1 is not remastered. The LOCKDIRWT parameter no longer controls any aspects of dynamic lock remastering. This parameter now only determines the likelihood of the node managing resource directory entries.

In a mixed-version cluster, the interaction between Version 8.3 nodes and nodes prior to Version 8.3 (which do not have a LOCKRMWT system parameter, still follows the old rules of using LOCKDIRWT. To avoid the situation of a lock tree constantly moving around the cluster, there is one exception to the above rule. A Version 8.3 master node never remasters a lock tree to a pre-Version 8.3 node if an interested node with a higher LOCKDIRWT value exists. This exception is necessary because the pre-Version 8.3 nodes immediately remaster the lock tree to the node with the higher LOCKDIRWT.

The SHOW CLUSTER utility is enhanced to display the LOCKRMWT system parameter for the nodes in the cluster. To view LOCKRMWT, use the command ADD RM_WT or ADD MEMBERS/ALL. For pre-Version 8.3 nodes, this field is displayed as **** to indicate these nodes do not have a LOCKRMWT system parameter.

3.5 Encryption for OpenVMS

OpenVMS Version 8.3 integrates the former Encryption for OpenVMS software product into the operating system. This eliminates the requirement for a separate product installation and product license. In addition, OpenVMS Version 8.3 now includes support for the Advanced Encryption Standard (AES) algorithm, which allows OpenVMS users, system managers, security managers, or programmers to secure their files, save sets, or application data with AES encryption.

Encryption is used to convert sensitive or otherwise private data to an unintelligible form called cipher text. This is done for the purpose of data confidentiality. Decryption reverses this process, taking the unintelligible cipher text and converting the data back into its original form, called plain text. Encryption and decryption are also known as encipher and decipher.

3.5.1 AES Features

AES encryption provides the following features and compatibility:

- The former data encryption standard (DES) algorithm is maintained for use with existing DES data and their applications. All the functions that existed with DES continue to provide that same level of DES support.
- AES encryption is integrated with BACKUP for encrypting and decrypting save sets with AES or with DES.
- Command-line use of AES encryption remains the same with only minor changes to qualifiers.
- ENCRYPT$ application program interface (API) changes are minimal with only textual parameter or flag changes required to use the AES algorithm.
- AES encryption supports the AES algorithm with four different cipher modes. With each mode, a secret key can be specified by the user in three different lengths (128, 192, and 256 bits), for a total of 12 different cipher and decipher operations:
  - AESCBC128 ! Cipher Block Chaining
  - AESCBC192 ! Cipher Block Chaining
  - AESCBC256 ! Cipher Block Chaining
  - AESECB128 ! Electronic Code Book
  - AESECB192 ! Electronic Code Book
  - AESECB256 ! Electronic Code Book
  - AESCFB128 ! Cipher Feedback
  - AESCFB192 ! Cipher Feedback
  - AESCFB256 ! Cipher Feedback
  - AESOFB128 ! Output Feedback
  - AESOFB192 ! Output Feedback
  - AESOFB256 ! Output Feedback

- These additional AES algorithm, modes, and key sizes are specified in the *algorithm* parameter to the ENCRYPT$ENCRYPT_FILE( ), and ENCRYPT$INIT( ) APIs or specified in the *algorithm-name* parameter for the ENCRYPT$GENERATE_KEY( ) API.

3.5.2 /CREATE_KEY /AES Command Qualifier

The AES keys (as well as DES keys) are created with the ENCRYPT command-line qualifier /CREATE_KEY. However, for AES keys, the /AES qualifier must be added:

```
$ ENCRYPT /CREATE_KEY keyname "This is my secret key" / AES
```

This currently generates an AES key with a key length of 21 characters. You can specify any length key as long as it meets the key-length minimum requirement and does not exceed Encrypt's maximum number of characters (approximately 240).

3.5.3 AES Key-Length Requirements

The AES key requirements are the actual number of bits used for each of the AES modes. This is actually the minimum number of bytes needed for the encryption or decryption operation. The minimum required key sizes are as follows:

- 128 bit mode = 16 byte key
- 192 bit mode = 24 byte key
- 256 bit mode = 32 byte key

3.5.4 Literal Key Values and ASCII Compression

Note that literal key values are not normally compressed and are passed as is to the encryption algorithm. Literal key values can be ASCII, HEX, or binary. However, ASCII DES key values are compressed if the descriptor data type is of data type DSCK_TYPE_T, DSCK_TYPE_VT or DSC$K_TYPE_VT. AES keys are not compressed. Other descriptor data types allow the DES literal key value to not be compressed and to pass through unchanged.

Literal keys values are specified at the command line with the /HEX qualifier or with the *literal* key flag to the ENCRYPT$DEFINE _KEY() routine. Literal key values can also be passed directly to the ENCRYPT$INIT() routine using the **key-type** argument and passing the value in the key-name descriptor. DES ASCII key values specified at the command line are subject to compression whether they are quoted or not quoted.

3.5.5 XOR Key Flag, or Key Folding

Encrypt will XOR any extra characters within a key to fold the bytes onto themselves, creating the number of bytes for the algorithm and key size for AES (or DES). This means that the maximum number of bytes (240) can be specified for a key value, but only 32 bytes are stored for AES and only 8 bytes are stored for DES when the key is created.

When this key is used, the original key is recovered, decrypted from storage. But only an 8-byte (folded) key are used for DES, and only 16-, 24-, or a 32-byte (folded) keys are used for AES, depending on the selected AES key size for the cypher operation.

The key size is specified as part of the *algorithm-name* parameter. For example, "AESCBC256" is specified to one of the following APIs or in the /DATA_ALGORITHM or /KEY_ALGORITHM qualifier to the file ENCRYPT and DECRYPT command:

- ENCRYPT$ENCRYPT_FILE( )
- ENCRYPT$INIT( )
- ENCRYPT$GENERATE_KEY( )

Creating an AES Key Example

The following example creates a 32-byte AES key. The key is encrypted with AES (currently AESCBC128) and is stored under the name ENCRYPT$KEY$MY_KEY in the process logical name table (by default). The key is flagged as an AES key to distinguish it from a DES key:

```
$  encrypt/create MY_KEY "This is a sample ASCII key value" /aes/log
```

```
%ENCRYPT-S-KEYDEF, key defined for key name = MY_KEY
```

3.5.6 ENCRYPT$DEFINE_KEY( ) API

AES and DES keys can also be created with the Encrypt application program interface (API), ENCRYPT$DEFINE_KEY( ). The key flags are used to distinguish the type of key (name or literal key value), and the logical name table to store the key.

There is also an AES key flag mask ENCRYPT$M_KEY_AES, and value ENCRYPT$V_KEY_AES, that is used to create an AES key.

```
ENCRYPT$DEFINE_KEY ( key-name , key-value , key-flags )
```

A random key value can be generated with the ENCRYPT$GENERATE_KEY( ) API.

```
ENCRYPT$GENERATE_KEY (algorithm-name , key-length
             [,factor-a] [,factor-b] [,factor-c]
             [,key buffer])
```

AES Key Flag

The following AES mask can be used in addition to (OR with) other flags for the key-flags parameter (as a longword by reference). An associated AES key value can be used for testing the bit within the program. Use the KEY_AES key flag to specify an AES key with the ENCRYPT$DEFINE_KEY( ), ENCRYPT$DELETE_KEY( ), and ENCRYPT$GENERATE_KEY( ) APIs:

- ENCRYPT$M_KEY_AES
- ENCRYPT$V_KEY_AES

3.5.7 Notes on Keys

The following list provides information about keys:

- AES keys are created, encrypted (always with AESCBC128 and a master key), and stored in a logical name table. During an encrypt operation, the key is fetched, decrypted, and used as a 16-, 24- or 32-byte key, depending on the chosen algorithm and key size for the encrypt or cypher operation.
- Nonliteral DES keys are compressed, that is, converted to uppercase. Only the characters A–Z, 0˙-9, dollar sign ($), period (.), and underscore (_) are allowed, all others are converted to spaces, and multiple spaces are removed. AES ASCII key values are not compressed.

- Care must be used when creating keys to ensure they meet the minimum key length when later used for the algorithm/key size selected. This was not a problem with 8-byte DES keys. Any key (literal or nonliteral) that is longer in length than necessary is folded for the proper 16-, 24- or 32-byte key size.
- The key name is a logical name for the key as stored in the logical name table (SYSTEM, JOB, GROUP, or PROCESS ˘- the default). The value can be ASCII (normal text keys), or hexadecimal/binary. When creating a literal key (*key-flags* = ENCRYPT$M_LITERAL_KEY), the value is stored as a literal value and it is not compressed.
- Care must also be used later when supplying the key to the ENCRYPT$INIT( ) API to match the key stored in the logical name table. That is, the descriptor type determines how the DES key is handled — as text to be compressed or as a binary value not to be compressed. AES key values are not compressed. The key flag (1= literal, 0=name) determines how the *key-name* parameter is interpreted; as a literal value passed directly to INIT, or key name for logical name lookup, translation and decryption. Note the errors that can result if an incorrect key type is used, for example, the key flag = 0 (name) and a literal key value is provided instead of a key name. An error could also occur trying to provide a key name to be used as a literal value. For the ENCRYPT$INIT( ) API, key name descriptors of type DSCK_DTYPE_T, DSCK_DTYPE_VT, and DSC$K_DTYPE_Z all specify that the key value should be compressed for DES keys. AES key values are not compressed.
- Errors can result when using the ENCRYPT$GENERATE_KEY( ) to generate AES keys and specifying key lengths that are not multiples of 16.

3.5.8 Deleting AES Keys

AES (and DES) keys are deleted or removed with the encrypt command-line qualifier /REMOVE_KEY or with the API ENCRYPT$DELETE_KEY( ):

```
$ ENCRYPT/REMOVE_KEY KEYNAME /AES
```

The user´s secret key is encrypted with a master key and stored in a logical name table (PROCESS, JOB, GROUP or SYSTEM—ENCRYP$SYSTEM table), the default is the PROCESS logical name table. To delete a key in a table other than the PROCESS logical name table, the appropriate qualifier (/JOB, /GROUP, or /SYSTEM) must also be specified in the ENCRYPT /REMOVE_KEY command.

Because the user´s secret key name is unique, only one key with the same name can exist in the same logical name table, whether this is a DES key or an AES key. This means that the /AES qualifier is unnecessary, although it is implemented nevertheless.

3.5.9 ENCRYPT$DELETE_KEY( ) API

To remove the key from the logical name table with the Encrypt API, specify the name of the key to be deleted. The flags specify which logical name table.

ENCRYPT$DELETE_KEY (*key-name* , *key-flags*)

AES Key Flag

The following AES mask can be used in addition to (OR with) other flags for the *key-flags* parameter (as a longword by reference). An associated AES key value can be used for testing the bit within the program. Use the KEY_AES key flag to specify an AES key with the ENCRYPT$DEFINE_KEY( ), ENCRYPT$DELETE_KEY( ), and ENCRYPT$GENERATE_KEY( ) APIs.

- ENCRYPT$M_KEY_AES
- ENCRYPT$V_KEY_AES

3.5.10 File Encryption and Decryption

Once a key has been created, a user can encrypt and decrypt files. This can be accomplished at the command line with the ENCRYPT and DECRYPT commands, or by using the ENCRYPT$ENCRYPT_FILE() API.

File encryption encrypts RMS files in fixed-length, 512-byte records. The file characteristics and attributes of the file are preserved, for example, the file creation and modify date, and whether the file was organized as sequential or indexed, and its record format (STREAM_LF, VAR, or other). A user specifies a key to be used for the encryption of a file and a data algorithm. But, the user's key is used to encrypt the random key, initialization vector (IV), and data algorithm in the random key record. Using the data algorithm specified by the user, it is the random key that encrypts the file's attributes and feature records and its data records, .

When decrypting the file, the key that the user specifies is used to decrypt the random-key record, which retrieves the random (data) key, IV, and data algorithm. Then the file attributes, feature records, and data records are decrypted with the random key, IV, and data algorithm from the fixed-length 512-byte records, and then restored to its original format and

3.5.10.1 File Encrypt and Decrypt Default Mode—DESCBC

By default, when encrypting a file from the command line, Encrypt uses the DESCBC algorithm to encrypt the file. That is, if a key or data algorithm is not specified on the command line, the DESCBC algorithm and mode is used.

An example that encrypts the file *file-name* using the key *key-name* to an output file with the file name of *file-name* using DESCBC is:

```
$ ENCRYPT file-name key-name
```

The following command is used to decrypt the file with DESCBC:

```
$ DECRYPT file-name key-name
```

3.5.10.2 Specifying the AES Data Algorithm and AES Key Algorithm

To select an algorithm other than the DESCBC default when encrypting files, Encrypt accepts the data and key algorithm qualifiers with the DCL ENCRYPT command and the key algorithm qualifier with the DECRYPT command.

When encrypting files with AES, specify both /DATA_ALGORITHM=AES*mmmkkk* and /KEY_ALGORITHM=AES*mmmkkk*:

- *mmm* defines the AES mode: ECB, CBC, CFB, or OFB
- *kkk* defines the key size: 128, 192, or 256 bits (for 16-, 24- or 32-byte keys)

Encrypt expects that the key matches the key algorithm. An AES key must be used with an AES key algorithm, and a DES key must be used with the DES key algorithm. Although the data algorithm will default to DES if the /DATA_ALGORITHM=AES*mmmkkk* is not specified for the ENCRYPT command. The same holds true when working with DES keys and KEY_ALGORITHM=DES, the data is protected with a strong algorithm, but the key is not.

Note

The capability of mixing AES with DES key and data algorithms has been disabled in OpenVMS Version 8.3, and any attempt to do so result in an ENCRYPT$_AESMIXDES error condition.

When decrypting files with AES, specify only the /KEY_ALGORITHM=AES*mmmkkk* qualifier. That is because the key algorithm is used to decrypt the random-key record that contains the random key that is then used to decrypt the data records of the file. Specifying the data algorithm is not necessary and, in fact, gives an unrecognized-qualifier error message.

Note

For an encrypt operation, if the /DATA_ALGORITHM=AES is specified without the /KEY_ALGORITHM, an error occurs. The default algorithm DESCBC is used to encrypt the random key record that contains the random key and file information. However, Encrypt expects that the user´s key matches the KEY algorithm; if not, an error occurs. That is, if the *key-name* is an AES key name and value, when the key is fetched from the logical name table and then is decrypted with the DES master key, the key decrypts garbage, and the operation fails with:

```
%STR-F-FATINTERR, fatal internal error
```

ENCRYPT /DATA_ALGORITHM=AES /KEY_ALGORITHM=AES

AES has a default encryption and decryption routine (AESCBC128) that is used when AES is specified without a mode and key size (that is, only /AES is specified). This could be used as a shortcut for AES file encryption. For example:

```
$ ENCRYPT file-name key-name /KEY=AES /DATA=AES
```

3.5.10.3 Specifying Only the Key Algorithm

To select an algorithm other than the DESCBC default when decrypting files, Decrypt accepts only the key algorithm qualifier with the DCL DECRYPT command. When decrypting with AES, specify only the /KEY_ALGORITHM=AESmmmkkk qualifier, where mmm defines the AES mode

Only the key algorithm needs to be specified. The data algorithm is stored with other file information in the encrypted file within a key record. The key record was encrypted with the user-specified encryption key when the file was encrypted. During the decrypt operation, the user´s key is used to decrypt the key record that contains the data key (a random key generated during the encrypt), along with its algorithm is then used to decrypt the remaining data records in the file.

3.5.11 ENCRYPT$ENCRYPT_FILE() API

AES File Flag

This is the command format for the ENCRYPT$ENCRYPT_FILE() API:

```
ENCRYPT$ENCRYPT_FILE(input-file, output-file,
                      key-name, algorithm, file-flags,
                      item-list )
```

There is an additional FILE_AES flag mask (and value) that is used with the ENCRYPT$ENCRYPT_FILE( ) API when encrypting files using an AES algorithm. The ENCRYPT$ENCRYPT_FILE_FLAGS are used to control file operations such as cipher direction, file compression and so on. The FILE_AES flag controls file AES initialization and encryption operations, and also to flag an AES key:

- ENCRYPT$M_FILE_AES
- ENCRYPT$V_FILE_AES

The optional item list is used to override the data algorithm parameter. The intent is to substitute one algorithm for another that is similar in function but different in name. You override the name of the algorithm in the random-key record with the name of the algorithm provided by the user in the override descriptor. This provides a way to open files that were encrypted with algorithm name that may be different than the algorithm name in the

3.5.12 Record Encryption/Decryption

File records can be encrypted and decrypted with the Encrypt API:

```
ENCRYPT$ENCRYPT_ONE_RECORD (input, output, key-name, algorithm)
```

```
ENCRYPT$DECRYPT_ONE_RECORD (input, output, key-name, algorithm)
```

To utilize AES for one record ciphers, an AES key must first be created, which is stored in the logical name table (encrypted). The key name of an AES key is specified and an address of a descriptor that contains the ASCII text for the selected AES*mmmkkk* (mode and key size) algorithm. Note that the input and output buffers (descriptor addresses) are also provided.

These one-record APIs assume that a key already exists in the logical name table key storage. These APIs are primarily used to encrypt and decrypt small amounts of data or only a few records. This is because there are overhead operations involved when calling the ENCRYPT$ENCRYPT_ONE_RECORD( ) API. Calling this API also calls the ENCRYPT$INIT(), ENCRYPT$ENCRYPT() or ENCRYPT$DECRYPT() function, and the ENCRYPT$FINI() function. An INIT, DECRYPT, and FINI function are also called recursively within the first INIT to decrypt the key in logical name storage, using the key name specified by the user in the *key-name* parameter.

HP recommends that you do *not* use the ENCRYPT$*xx*CRYPT_ONE_RECORD() calls if many records need to be enciphered or deciphered. Instead, HP recommends that the ENCRYPT$ENCRYPT() and ENCRYPT$DECRYPT() API functions be used for normal operations. This implies that the ENCRYPT$INIT() function is also used to initialize the context prior to encryption or decryption, and that the ENCRYPT$FINI() API is used to free memory structures prior to application exit.

3.5.13 Data Encryption/Decryption

The ENCRYPT$ENCRYPT() and ENCRYPT$DECRYPT() routines are used by applications to cryptographically process up to 64K bytes of data.

```
ENCRYPT$ENCRYPT (context, input, output [,output-length] [,initialization-vector]
ENCRYPT$DECRYPT (context, input, output [,output-length] [,initialization-vector]
```

These routines require initializing an encryption context with ENCRYPT$INIT() routine, prior to calling ENCRYPT$ENCRYPT() or ENCRYPT$DECRYPT() to encipher or decipher the data blocks. The ENCRYPT$FINI() routine is called at the end to free the context data structures.

The output buffer must be able to accommodate a padded block to an increment of the block length. For AES, this is 16 bytes (8 bytes for DES). The *output-length* value and *initialization-vector* (IV) parameter are optional. The *output-length* is the number of bytes written (encrypted or decrypted).

The AES IV is a reference pointer to a 16-byte value. Internal structures have been expanded to accommodate AES. The DES IV is a quadword reference to an 8-byte value.

3.5.14 Lengths and Block Mode Padding

The AES block-mode algorithms (AESCBC*xxx* and AESECB*xxx* pad the data to even 16-byte block boundaries. For AES, 1 byte encrypts and decrypts to 16 bytes, 72 bytes to 80, and so forth. The AES padding character is a hexadecimal number of bytes indicating the number of bytes padded. For example, the 1-byte encrypt pad would be 15 characters of 0F following the 1 encrypted byte of data (08 08 ... 08) of 8 bytes following the 72 bytes of data. DESECB and DESCBC modes always pad with characters of zeros. The character stream modes (AESCFB*xxx*, AESOFB*xxx*, DESCFB) do not pad the data and the *output-length* parameter matches the actual number of bytes.

3.5.15 New AES Encryption Key, Flag Mask, and Value

There are no new AES encryption API routines. However, to accommodate the AES algorithm and the various key-length values, an additional AES key and AES file flag mask and value are added to OpenVMS Version 8.3.

- AES key flag

  The KEY_AES mask value specified an AES key (as a longword by reference) to the ENCRYPT$DEFINE_KEY( ), ENCRYPT$DELETE_KEY( ), and ENCRYPT$GENERATE_KEY( ) APIs:

  — ENCRYPT$M_KEY_AES

  — ENCRYPT$V_KEY_AES

- AES file flag

  An additional FILE_AES flag mask (and value) is used with the ENCRYPT$ENCRYPT_FILE( ) API when encrypting files that use an AES algorithm.

  The ENCRYPT$ENCRYPT_FILE_FLAGS flags are used to control file operations such as cipher direction, file compression, and so on. The FILE_AES flag controls file AES initialization and encryption operations and also flags the following AES keys:

  - ENCRYPT$M_FILE_AES
  - ENCRYPT$V_FILE_AES

The AES algorithm, mode, and a key length (128, 192, or 256 bits) are specified in the *algorithm* parameter for the ENCRYPT$ENCRYPT_FILE( ), and ENCRYPT$INIT( ) APIs, or the are specified in the *algorithm-name* parameter for the ENCRYPT$GENERATE_KEY( ) API. This parameter is in the form of a character string descriptor reference (pointer), as follows:

- Block mode ciphers
  - AESCBC128 ! Cipher Block Chaining
  - AESCBC192 ! Cipher Block Chaining
  - AESCBC256 ! Cipher Block Chaining
  - AESECB128 ! Electronic Code Book
  - AESECB192 ! Electronic Code Book
  - AESECB256 ! Electronic Code Book
- Stream mode ciphers
  - AESCFB128 ! Cipher Feedback
  - AESCFB192 ! Cipher Feedback
  - AESCFB256 ! Cipher Feedback
  - AESOFB128 ! Output Feedback
  - AESOFB192 ! Output Feedback
  - AESOFB256 ! Output Feedback

Note

AESCBC128 is the default cipher and is also used for encryption and decryption of the user´s key for storage of logical names. These ciphers are searched in the order in which they are stored in their algorithm table (as listed) within the new image file SYS$SHARE:ENCRYPT$ALG$AES.EXE file.

3.5.16 Unsupported AES Encryption Operations

The following AES encryption operations are not supported and, therefore, are not recommended:

- Message Authentication Code (MAC)

  The Message Authentication Code (MAC) detects any modifications made to a file´s data or to its security settings. Currently, only DES is supported for MAC operations. AES is not supported.

  The MAC is used with the /AUTHENTICATE command qualifier. The MAC encrypts the file´s data (and security attributes), storing then in two separate databases (Db). To detect file modifications, the MAC is recalculated and compared with the Db MAC.

The authentication codes are generated with the /UPDATE qualifier, and are logged or displayed with the /OUTPUT=*file-name* qualifier. For example:

```
$ encrypt /AUTHENTICATE /UPDATE *.exe KeyName /out=tt:
```

The MAC also uses an IV, but DESCBC is the underlying algorithm and mode for the keyed file MAC. The MAC is the final DESCBC encrypted block of the file´s data, the file´s security attributes.

- ENCRYPT/COMPRESS

  Using ENCRYPT/COMPRESS with BACKUP file save sets is not recommended, because of errors encountered during their decryption. This usually happens with larger save sets created with a /GROUP_SIZE not equal to zero.

  ENCRYPT/COMPRESS works properly but decryption might fail. This can be catastrophic if the /DELETE qualifier is used, deleting the original BACKUP save-set file during the encrypt operation.

- Encrypting files with AES

  Specify both the /DATA=AES*mmmkkk* and the /KEY=AES*mmmkkk* algorithm when encrypting AES files, where *mmm* is the mode (CBC,ECB,CFB, or OFB) and *kkk* is the key size (128, 192, or 256 bits).

- Mixing AES and DES keys and algorithms

  Encrypt expects that the key matches the key algorithm. An AES key must be used with an AES key algorithm, and a DES key must be used with the DES key algorithm. The data algorithm can default to DES if you specify an AES /KEY_ALGORITHM qualifier together with an AES key without specifying AES for the /DATA_ALGORITHM qualifier. For security reasons, we disallow this by signaling an ENCRYPT$_AESMIXDES error at the command line. The same holds true when working with DES keys and KEY_ALGORITHM=DES; the data is protected with a strong algorithm but the key is not. This command-line capability of mixing key and data algorithms between DES and AES has been disabled in OpenVMS 8.3. Note that other errors can result when mixing AES and DES keys and algorithms.

3.6 Monitor Utility Enhancements

The following sections describe enhancements to the Monitor utility.

3.6.1 Align Command (I64 Only)

The Monitor utility has been enhanced to display information about alignment faults. This new MONITOR ALIGN command is valid only on OpenVMS for Integrity servers and helps troubleshoot performance problems on Integrity server systems.

The MONITOR ALIGN class displays a rate of alignment faults for each mode (kernel, executive, supervisor and user) along with the total alignment faults per second. If the alignment fault rate per second is very high, use the Alignment Fault utility (FLT), which is run through SDA, to analyze the cause of the alignment faults.

On Integrity server systems, all alignment faults are handled by the operating system, hence counters can be incremented to track the alignment fault rate. On Alpha, alignment faults are fixed in PALcode in the console so counters cannot be ticked without much overhead. For this reason, the MONITOR ALIGN command is only available on Integrity servers.

The header include file $MONDEF has also been enhanced to include the record definitions for the new ALIGN class. Previously, the constants for each class type record have not been provided, but in Version 8.3 $MONDEF also includes the symbolic constant definitions for the class numbers as MNR_CLS$K_xxx.

Note the following example:

```
$ monitor align
                         ALIGNMENT FAULT STATISTICS
                               on node MTDIB9
                          11-JAN-2006 16:58:07.25

                                  CUR         AVE         MIN         MAX

    Kernel Alignment Faults       19529.00    19529.00    19529.00    19529.00
    Exec Alignment Faults          7581.00     7581.00     7581.00     7581.00
    Super Alignment Faults            0.00        0.00        0.00        0.00
    User Alignment Faults        164972.00   164972.00   164972.00   164972.00

 Total Alignment Faults       192082.00   192082.00   192082.00   192082.00
```

3.6.2 New Classname Qualifiers for the PROCESSES Class

The four new classname qualifiers for the MONITOR utility PROCESSES class can be used to monitor per-process-based modes usage. They are useful in helping to identify the top consumers of the various CPU modes. If, for example, the MONITOR MODES command shows that an excessive amount of supervisor mode is being used, the new MONITOR PROCESSES/TOPSUPERVISOR display will reveal which process—and hence, which user—is responsible.

The new qualifiers are described in the following table:

Table 3–2 MONITOR utility Classname Qualifiers for the PROCESSES Class

Command and Qualifier	Description
MONITOR PROCESSES /TOPKERNEL	Top kernel mode usage per process
MONITOR PROCESSES /TOPEXECUTIVE	Top executive mode usage per process
MONITOR PROCESSES /TOPSUPERVISOR	Top supervisor mode usage per process
MONITOR PROCESSES /TOPUSER	Top user mode usage per process

See the *HP OpenVMS System Management Utilities Reference Manual*, in the MONITOR chapter, for more detailed information.

3.6.3 MONITOR PROCESSES/TOPSUPERVISOR Example

The new MONITOR PROCESSES/TOPSUPERVISOR qualifier allows you to see which processes are top consumers of CPU in supervisor mode. For more information about these qualifiers, see the *HP OpenVMS System Management Utilities Reference Manual*.

The following example will appear in the section “Using Live Display Monitoring” in the chapter “Getting Information About the System” in the next version of the *HP OpenVMS System Manager’s Manual*.

Example:

```
$ MONITOR PROCESSES/TOPSUPERVISOR
```

This command displays a bar graph showing the 16 processes that are the top consumers of CPU time in supervisor mode. Values are expressed in units of clock ticks (10ms) per second.

The command produces a display similar to the following:

```
                           OpenVMS Monitor Utility
                       TOP SUPERVISOR MODE PROCESSES
                             on node QUEBIT
                         7-DEC-2005 14:04:24.19

                             0         25        50        75       100
                             + - - - - + - - - - + - - - - + - - - - +
 74E000AD  BATCH_3   5  **
 74E000AC  BATCH_2   4  *
 74E000AA  BATCH_1   3  *
 74E000AB  _RTA3:    3  *
```

3.7 Multipath Enhancement for Active-Active Feature of EVA and MSA Controllers

The controllers of the Enterprise Virtual Array (EVA) 4000/6000/8000 storage systems and of the MSA1500 storage system provide “active optimized” (AO) and “active non-optimized” (ANO) paths. This feature will also be offered in the EVA 3000/5000 storage systems. There is a read I/O performance penalty for using the ANO paths.

The OpenVMS multipath capability has been enhanced to distinguish between the AO and the ANO paths to improve I/O performance. Users should notice a performance improvement, which will vary depending on I/O size and queue depth. The longer the queue depth, the greater the improvement users will notice.

For more information about the OpenVMS multipath capability, refer to *Guidelines for OpenVMS Cluster Configurations*. For more information about the controllers on these storage systems, visit:

`http://www.hp.com/country/us/en/prodserv/storage.html`

For more information about the EVA 4000/6000/8000 controllers, select Browse by capacity, Enterprise, and select the storage system of interest.

Similarly, for more information about the EVA 3000 and 4000 controllers, select Browse by capacity, Mid-range, and select the storage system of interest. For more information about the MSA 1500 controller, select Browse by capacity, Entry-level, MSA 1500.

3.8 OpenVMS Cluster Interconnect

The following features are added to the LAN-based cluster communications driver (PEdriver) in in OpenVMS Version 8.3:

- Data compression
- Multi-gigabit line speed & long distance performance scaling

Data compression may be used to reduce the time to transfer data between two OpenVMS nodes when the LAN speed between them is limiting the data transfer rate, and there is idle CPU capacity available. For example, it may be used to reduce shadow copy times, or improve MSCP serving performance between Disaster Tolerant Clusters sites connected by relatively low-speed links such as E3 or DS3, FDDI, or 100Mb Ethernet. PEdriver data compression can be enabled by using SCACP, Availability Manager, or the NISCS_PORT_SERV sysgen parameter.

The number of packets in flight between nodes needs to increase proportionally to both the speed of LAN links and the inter-node distance. Historically, PEdriver had fixed transmit and receive windows (buffering capacity) of 31 outstanding packets. Beginning with OpenVMS Version 8.3, PEdriver now automatically selects transmit and receive window sizes (sometimes called pipe quota by other network protocols) based on the speed of the current set of local and remote LAN adapters being used for cluster communications between nodes. Additionally, SCACP and Availability Manager now provide management override of the automatically-selected window sizes.

For more information, see the SCACP utility chapter, and NISCS_PORT_SERV in the *HP OpenVMS System Management Utilities Reference Manual* and the *HP OpenVMS Availability Manager User's Guide*.

3.9 OpenVMS Operating System Media Patch-Related Menu Option

The OpenVMS operating system distribution media main menu now includes a new option (7) that enables you to perform patch-related operations. When you select option 7, you are brought to a submenu that provides options enabling you to search the patch kits, install patches, remove recent patches for which there is recovery data, and to show and delete recovery data. As a result, you can perform these operations even when the operating system cannot be booted (in which case you cannot use the PCSI PRODUCT command). This example shows the menu options:

```
Please choose one of the following:

  1) Upgrade, install or reconfigure OpenVMS I64 Version X8.3-BBV
  2) Display layered products that this procedure can install
  3) Install or upgrade layered products
  4) Show installed products
  5) Reconfigure installed products
  6) Remove installed products
  7) Find, Install, or Undo patches; Show or Delete Recovery Data
  8) Execute DCL commands and procedures
  9) Shut down this system

Enter CHOICE or ? for help: (1/2/3/4/5/6/7/8/9/?)
```

Although options 2 and 3 of the main menu can still be used to perform patch operations, HP recommends that you use the new option 7 submenu operations; the functionality provided by these operations is more extensive and more reliable. For example, when you install patches using the new submenu option, recovery data is saved automatically. In addition, the new options allow you to specify locations on which to perform the operations (in addition to the default target device). You can use wildcards to specify locations.

3.10 PCSI Utility Enhancements

OpenVMS Version 8.3 contains the following enhancements for the PCSI utility:

- Manual verification of the product database using the PRODUCT ANALYZE PDB command
- Automatic verification of the product database using the PRODUCT INSTALL command

- Full support for ODS-5 volumes
- Validation of signed product kits performed by several PRODUCT commands

3.10.1 PRODUCT ANALYZE PDB

The new PRODUCT ANALYZE PDB command verifies the structural integrity of the product database and, in some circumstances, performs minor repairs. This command:

- Reads all SYS$SYSTEM:*.PCSI$DATABASE files that are referenced in the root file PCSI$ROOT.PCSI$DATABASE.
- Checks all fields in these files for correct syntax.
- Automatically performs minor repair when a known corruption pattern can be identified and a repair is feasible.
- Provides instructions on how to rebuild the database if an unrecoverable corruption is found.

For information on optional qualifiers for the command, see the POLYCENTER Software Installation utility chapter in the *HP OpenVMS System Management Utilities Reference Manual*.

3.10.2 Automatic Verification of the Product Database

The PRODUCT INSTALL command has been enhanced to automatically verify the product database at the start of the operation and again when the product database is updated. The verification and repair actions are similar to those provided by the PRODUCT ANALYZE PDB command. Other PRODUCT commands that modify the database (such as PRODUCT RECONFIGURE and PRODUCT REMOVE) also perform a verification pass.

3.10.3 Support for ODS-5 Volumes

The PCSI utility now provides full support for enhanced ODS-5 directory and file specification syntax. Prior to OpenVMS Version 8.3, product kits could be installed on ODS-5 disks, but all files were placed on the destination volume using ODS-2 syntax rules (for example, file names were in upper case only). With ODS-5 support, the kit developer can create a kit that installs files utilizing enhanced syntax such as:

- Upper, lower, and mixed case names
- Multiple dots in file specifications
- Extended character set
- Long names

The technique used to specify enhanced syntax is to quote the name specification parameter of the DIRECTORY or FILE statement in the product description file (PDF). For example, consider a PDF file that contains the following two lines:

```
file [test]file_one.txt ;
file "[test]File_Two.txt" ;
```

If the destination disk is an ODS-5 volume, the PCSI utility for OpenVMS Version 8.3 will install the first file as FILE_ONE.TXT and the second file as File_Two.txt. For compatibility with older versions of the PCSI utility, an unquoted specification is changed to all uppercase characters.

In addition, the product developer can test whether a volume is ODS-2 or ODS-5 by use of a new PDF statement of the form:

```
IF ( < FILESYSTEM { ODS-2 | ODS-5 } [ VOLUME { DESTINATION | SYSTEM } ] > ) ;
```

For example, to conditionally process statements if the destination volume is ODS-5, the following lines can be used:

```
IF ( < FILESYSTEM ods-5 VOLUME destination > ) ;
...
END IF ;
```

3.10.4 Support for Secure Delivery of Product Kits

Several PRODUCT commands have been enhanced to support Secure Delivery of product kits. See Section 5.2 for an overview of Secure Delivery. The following capabilities have been added to the PCSI utility:

- All PRODUCT commands that read product kits will validate signed kits before use if their associated digital signature file (also referred to as a manifest) is present in the source directory. These commands are PRODUCT CONFIGURE, COPY, EXTRACT, INSTALL, LIST, RECONFIGURE, and REGISTER PRODUCT. A digital signature file has the same file name and file type as the product kit with an "_ESW" appended to the file type. For example, HP-I64VMS-TEST-V0100–1.PCSI$COMPRESSED_ESW.
- The new PCSI utility can install unsigned kits and older versions of the PCSI utility can install signed kits except that they will not be validated.
- Kit validation can be disabled by use of the /OPTIONS=NOVALIDATE_KIT qualifier.
- The PRODUCT COPY command will copy both the specified kits and their manifest files (if present).
- The PRODUCT SHOW HISTORY command has a new field called VAL which indicates the validation status of a product. Possible status codes are:
  - VAL — kit was successfully validated
  - SYS — kit was installed from OS media during an OpenVMS installation or upgrade, but not validated
  - (U) — kit was not validated because it was unsigned
  - (M) — kit was not validated because no manifest file was found in the kit's source directory
  - (D) — kit was not validated because validation was disabled by user request
  - (C) — kit was not validated because CDSA was not operational
  - <hyphen> — not applicable for the operation (such as removal of a product)
  - <blank> — operation was performed by a version of the PCSI utility that does not support secure delivery
- The output from the PRODUCT LIST command contains additional information such as whether the kit is signed and a manifest file has been found.

For more information about PRODUCT commands and qualifiers that support secure delivery, see the POLYCENTER Software Installation utility chapter in the *HP OpenVMS System Management Utilities Reference Manual*.

3.10.5 Defaults Changed on Two Qualifiers

In OpenVMS Version 8.3, default values for certain qualifiers are changed for the following commands:

- PRODUCT INSTALL

  The following defaults are changed:

  — The /RECOVERY_MODE qualifier is now the default when you install a product. (This is a change from /NORECOVERY_MODE as the default.)

  — The /SAVE_RECOVERY_DATA qualifier is now the default when you install patch and mandatory update kits. (This is a change from /NOSAVE_RECOVERY_DATA as the default.)

- PRODUCT RECONFIGURE

  The /RECOVERY_MODE qualifier is now the default when you reconfigure a product. (This is a change from /NORECOVERY_MODE as the default.)

3.11 SANCP Utility

The SANCP utility allows you to limit the number of active I/Os a host may have across all paths to Logical Unit Numbers (LUN) on a given Fibre Channel storage port. A storage port can be selected by a discrete or wildcarded port World Wide ID (WWID) or by a product ID substring.

The SANCP utility processes a command and qualifiers passed to it on the command line, allowing it to be executed from a DCL script, or it prompts you if started with no command. For more information about the SANCP utility, see the SANCP Help facility.

3.12 SAS Utility (I64 Only)

The SAS utility (SAS$UTIL)is an OpenVMS system management and diagnostic tool that is capable of configuring Integrated RAID (IR) functionality for the HP 8 Internal Port Serial Attached SCSI Host Bus Adapter (SAS Controller).

Integrated RAID (IR) is used where extra performance, storage capacity, or redundancy of a RAID configuration, or all three, are required. OpenVMS Version 8.3 supports only Integrated RAID 1 or Integrated Mirroring (IM) and its associated Global Hot Spare capability.

For more detailed information, see the *HP OpenVMS System Management Utilities Reference Manual*.

3.13 SCACP utility

The following new features are added to the SCACP utility in OpenVMS Version 8.3:

- Data compression
- Multi-gigabit & distance performance scaling

These new features are described in the following sections.

3.13.1 Data Compression Management

The SCACP SET VC command now includes a /COMPRESSION (or /NOCOMPRESSION) qualifier, which enables or disables sending compressed data by the specified PEdriver VCs. The default is /NOCOMPRESSION.

You can also enable the VC use of compression by setting bit 3 of the NISCS_PORT_SERV system parameter. The /NOCOMPRESSION qualifier does not override compression enabled by setting bit 2 of NISCS_PORT_SERV.

3.13.2 Multi-Gigabit Scaling

You can use the SET VC COMMAND /WINDOW=RECEIVE_SIZE=*value* and /WINDOW=TRANSMIT_SIZE=*value* qualifiers to override the automatically calculated receive and transmit window sizes for a PEdriver VC. The /WINDOW=NORECEIVE_SIZE and /WINDOW=NOTRANSMIT_SIZE qualifiers can be used to remove management override of the window sizes. You can use /WINDOW=(NORECEIVE_SIZE,NOTRANSMIT_SIZE) to remove the override from both transmit and receive window sizes with a single command.

These new command qualifiers can be used to ensure that the VC has enough buffering to receive, and can transmit sufficient packets before waiting for return acknowledgments, to attain maximum bandwidth between nodes. To avoid unnecessary VC closures, these commands have restrictions on the order that they are issued. For information about these restrictions, see the *HP OpenVMS System Management Utilities Reference Manual* and the SCACP Help facility.

The new SCACP command, CALCULATE WINDOW_SIZE, can be used to determine the maximum window size that the VC should be using. This command has two required qualifiers, /DISTANCE=KILOMETERS (or MILES)=*n* and /SPEED=*s*, where *n* is the cable route distance between the two nodes, and *s* is the total bandwidth of all links being used for cluster communications between the nodes.

The SCACP SHOW VC command now displays if compression is enabled on a VC, and there are new columns for the management settings for transmit and receive window size.

For more information, see the SCACP utility chapter, and NISCS_PORT_SERV in the *HP OpenVMS System Management Utilities Reference Manual* and *HP OpenVMS Availability Manager User's Guide*.

3.14 HP OpenVMS I64 Serial Multiplexer (MUX) Support (I64 Only)

RS232 serial lines and multiplexers are used for a variety of tasks, from traditional terminal connections to low-speed system-to-system communications and even communications with remote instruments. OpenVMS has traditionally supported adding serial lines at the same time as option-card-based multiplexers. This solution requires dedicating I/O slots; it also limits the choices of option cards available.

With the widespread adoption of the Universal Serial Bus (USB) on industry-standard platforms, OpenVMS has moved away from option-card-based multiplexers and has adopted USB to add serial lines to HP Integrity servers. Rather than using one or two option-card solutions with 8 or 16 lines for all configurations, you can now configure USB to meet your exact requirements.

Testing shows that the USB-based serial multiplexers perform as well as (or better than) their option-card counterparts and cause very low overhead to the system. In fact, the overhead is lower than option-card-based multiplexers.

For more information about HP MUX support, see the *HP OpenVMS System Management Utilities Reference Manual*.

3.15 Spinlock Trace Utility (SPL)

The Spinlock Trace utility available through SDA has been changed to report the various spinlock hold and wait times in nanoseconds instead of cycles. This allows for easy comparison of information collected among systems with different cycle counter frequencies.

The display has also been enhanced to show the full 64-bit address of PCs as we are moving code to run in P2 and S2 space. Also the decoding of the PC address has been enhanced to show the module, routine and offset for I64. The SPL ANALYZE and SPL SHOW COLLECT commands displays the additional PC decoding by default, but SPL SHOW TRACE [/SUMMARY] shows the information only if the /FULL qualifier is specified because the display is already congested.

3.16 HP OpenVMS System Analysis Tools

The following sections describe the new features provided in the System Analysis Tools utilities. The *HP OpenVMS System Analysis Tools Manual* is not updated for this release, but the additions and changes noted in this manual and in the *HP OpenVMS Version 8.3 Release Notes* have been included in online help for the SDA utility and in related commands for ANALYZE and System Service Logging.

3.16.1 System Dump Debugger

The System Dump Debugger (SDD) is now supported on OpenVMS for Integrity servers as well as on OpenVMS Alpha.

3.16.2 System Dump Analyzer

The following sections describe the following new SDA or SDA extension commands and new callable routine extensions, as well as several new qualifiers for SDA commands:

- COLLECT command
- SHOW CLASS command
- SHOW EFI command
- SHOW VHPT command
- VALIDATE POOL command
- VALIDATE PROCESS command
- CLUE REGISTER
- CLUE SCSI
- SDA$CBB_BOOLEAN_OPER routine
- SDA$CBB_CLEAR_BIT routine
- SDA$CBB_COPY routine
- SDA$CBB_FFC routine
- SDA$CBB_FFS routine
- SDA$CBB_INIT routine

- SDA$CBB_SET_BIT routine
- SDA$CBB_TEST_BIT routine
- SDA$DELETE_PREFIX routine
- SDA$FID_TO_NAME routine
- SDA$GET_FLAGS routine

The Common Bitmask Block (CBB) routines, SDA$CBB_*xxx*, are designed for use with local copies of the CBB structures that describe the CPUs in use in a system. The CBB structures are assumed to be at least CBB$K_STATIC_BLOCK bytes in length. The definitions of the various CBB constants and field names used by these routines can be found in CBBDEF.H in SYS$LIBRARY:SYS$LIB_C.TLB.

The set of routines is not intended to be an exhaustive set of all possible CBB-related operations, but provides those operations known to be needed. They may not work as expected with CBB structures set up for any purpose other than to describe CPUs.

COLLECT

Collects file identification to file name translation data on both OpenVMS Alpha and OpenVMS for Integrity servers, and processes unwind data only on OpenVMS for Integrity servers.

Format

COLLECT [qualifiers]

Parameters

None

Qualifiers

/LOG
Displays information on the progress of the COLLECT command, for example, the name of the process being scanned, or (on Integrity servers) the name of an image whose unwind data is being collected.

/SAVE [= file-name]
Writes collection data to a separate file. By default, a file of type .COLLECT with the same name as the dump file is created in the same directory as the dump file.

/UNDO
Removes all the file or unwind data from an earlier COLLECT command from SDA's memory. COLLECT/UNDO does not affect the file or unwind data already appended to the dump file being analyzed, or already written to a separate collection file.

Description

When a dump is being analyzed, it is useful to have data available that cannot be written to the dump file at the time of the system crash. This data includes the full file specification associated with a file identification. On OpenVMS for Integrity servers, it also contains the unwind data for images activated in processes.

If the dump is being analyzed on the system where it is originally written, this data can be collected for use in the current SDA session using the COLLECT command. If the dump is being copied for analysis elsewhere, the COPY/COLLECT command can be used to collect the data and append it to the copy being written. If the COPY/COLLECT command is used after a COLLECT command, the data already collected is appended to the dump copy.

For all file or unwind data to be collected successfully, all disks that were mounted at the time of the system crash should be remounted and accessible to the process running SDA.

If the COPY and the COLLECT cannot be done as a single step, a COLLECT/SAVE command writes the collection to a separate file that can be used later with the dump file. A subsequent COPY command combines the two files.

SHOW CLASS

Displays information about scheduling classes that are active in the system or dump being analyzed.

Format

SHOW CLASS [class-name | /ALL]

Parameters

class-name
Name of the class to be displayed.

Qualifiers

/ALL
Indicates that details of all active classes are to be displayed.

Description

SDA displays information about active scheduling classes in the system. By default, a summary of the classes is displayed.

SHOW EFI (I64 Only)

Displays information from the Extensible Firmware Interface (EFI) data structures. Currently, the only display provided by SDA is the EFI memory map.

Format

SHOW EFI /MEMMAP [=ALL] [*range*]

Parameters

range

The entry or range of entries to be displayed, expressed using the following syntax:

m—Displays the entry *m*.
m:n—Displays the entries from *m* to *n*.
m;n—Displays *n* entries starting at *m*.

You cannot specify a range with /MEMMAP=ALL.

Qualifiers

/MEMMAP [=ALL]

Displays the EFI memory map. This qualifier is required. By default, only entries in the EFI memory map with the **runtime** attribute are displayed. If the /MEMMAP=ALL qualifier is specified, all entries are displayed.

You cannot specify the /MEMMAP=ALL qualifier and supply a range of entries to be displayed.

Description

SDA locates the EFI memory map in the system or dump and displays the contents. If no range is given, SDA also displays information about the location and size of the memory map.

SHOW VHPT (I64 Only)

Displays data from the Virtual Hash Page Table.

Format

SHOW VHPT [/CPU = { *n* | * } [/ALL] [*range*]]

Parameters

range
The entry or range of entries to be displayed, expressed using the following syntax:

m—Displays the VHPT entry *m*.
m:n—Displays the VHPT entries from *m* to *n*.
m;n—Displays *n* VHPT entries starting at *m*.

A range can only be provided if a single CPU is specified with the /CPU qualifier.

Qualifiers

/CPU = { *n* | * }
Indicates that the detailed contents of the VHPT for one or all CPUs is to be displayed. The default action is for a summary of VHPT information to be displayed.

/ALL
Displays all VHPTs for the specified CPU(s). Without /ALL, only entries that have a valid tag are displayed.

Description

Displays contents of the Virtual Hash Page Table on an OpenVMS I64 system. By default, a summary of the VHPT entries is displayed. If CPUs are specified, details of individual VHPT entries are displayed for the CPUs. If a single CPU is specified, specific VHPT entries for that CPU are displayed.

In the detailed display, the columns are as follows:

Table 3–3

Entry	VHPT Entry Number
Bits	One or more of the following flags: P—Present A—Accessed D—Dirty E—Exception deferral I—Tag invalid (only seen if /ALL is specified)
MA	One of the following memory attributes: WB—Write Back UC—Uncacheable UCE—Uncacheable Exported WC—Write Coalescing NaT—NaTPage
AR/PL	The access rights and privilege level of the page. Consists of a number (0-7) and a letter (K, E, S, or U) that determines access to the page in each mode.
KESU	The access allowed to the page in each mode. This is an interpretation of the AR/PL values in the previous column. For an explanation of the access codes, see *HP OpenVMS System Analysis Tools Manual*.
Physical address	The starting physical address for this VHPT entry.
Page size	The size of the page represented by this VHPT entry. Page sizes for VHPT entries range from 4KB to 4GB. Not all possible pages sizes are used by OpenVMS for Integrity servers.
Tag	The translation tag for the VHPT entry.
Quad4	Information recorded by OpenVMS for Integrity servers for debugging purposes. The contents of this quadword are subject to change.

VALIDATE POOL

Checks all free pool packets for POOLCHECK-style corruption, using the same algorithm as the system pool allocation routines when generating a POOLCHECK bugcheck and system dump.

Format

```
VALIDATE POOL   { /ALL (d) | /BAP | /NONPAGED | /PAGED }
                [ /HEADER | /MAXIMUM_BYTES [ = n ] /SUMMARY ]
```

Parameters

None

Qualifiers

/ALL
Checks free packets for all pool types (nonpaged pool, paged pool, and bus-addressable pool). This is the default.

/BAP
Checks free packets in bus-addressable pool.

/HEADER
Displays only the first 16 bytes of any corrupted free packets found.

/MAXIMUM_BYTES[=*n*]
Displays only the first *n* bytes of any corrupted free packets found. If you specify /MAXIMUM_BYTES without a value, the default is 64 bytes.

/NONPAGED
Checks free packets in nonpaged pool.

/PAGED
Checks free packets in paged pool.

/SUMMARY
Displays only a summary of corrupted pool packets found.

Description

The VALIDATE POOL command displays information about corrupted free pool packets. It is useful only if pool checking is enabled using either the POOLCHECK or the SYSTEM_CHECK system parameters. (For information about these system parameters, see the *HP OpenVMS System Management Utilities Reference Manual*).

VALIDATE PROCESS

Performs validation of process data structures. Currently, the only validation available is to check free process pool packets for POOLCHECK-style corruption, using the same algorithm as the system pool allocation routines when generating a POOLCHECK bugcheck and system dump.

Format

```
VALIDATE PROCESS/POOL   [ = { P0 | P1 | IMGACT | ALL (d) } ]
                        [ /ADDRESS = pcb-address | process name | ALL
                        | /ID = nn | /INDEX = nn | /NEXT | /SYSTEM ]
                        [ /HEADER | /MAXIMUM_BYTES [ = n ]
                        /SUMMARY ]
```

Parameters

ALL
Indicates that all processes in the system are to be validated.

process-name
Name of the process to be validated. The process name can contain up to 15 uppercase letters, numerals, an underscore (_), dollar sign ($), colon (:), and some other printable characters. If it contains any other characters (including lowercase letters), you might need to enclose the process name in quotation marks (" ").

Qualifiers

/ADDRESS = *pcb-address*
Specifies the process control block (PCB) address of the process to be validated.

/HEADER
Displays only the first 16 bytes of any corrupted free packets found.

/ID = *nn*
/INDEX = *nn*
Specifies the process to be validated by its index into the system's list of software PCBs, or by its process identification. You can supply the following values for *nn*:

- Process index itself.
- Process identification (PID) or extended PID longword, from which SDA extracts the correct index. The PID or extended PID of any thread of a process with multiple kernel threads can be specified. Any thread-specific data displayed by further commands is for the given thread.

To obtain these values for any given process, issue the SDA command SHOW SUMMARY/THREADS. The /ID=*nn* and /INDEX=*nn* qualifiers can be used interchangeably.

/MAXIMUM_BYTES[=*n*]
Displays only the first *n* bytes of any corrupted free packets found. If you specify /MAXIMUM_BYTES without a value, the default is 64 bytes.

/NEXT

Causes SDA to locate the next process in the process list and validate that process. If there are no further processes in the process list, SDA returns an error.

/POOL[= { P0 | P1 | IMGACT | ALL (d) }]

Process pool validation is to be performed. This qualifier is required. Use of a keyword on the /POOL qualifier allows the user to specify which process pool is to be validated (P0, P1, or Image Activator Pool). The default is to validate all process pools.

/SUMMARY

Displays only a summary of corrupted pool packets found.

/SYSTEM

This qualifier is provided for compatibility with SET PROCESS/SYSTEM and SHOW PROCESS/SYSTEM. There is no pool associated with the system process that can be validated. SDA sets its current process context to the system process and outputs the following text:

```
Options ignored for System process:  POOL
```

Description

The VALIDATE PROCESS command validates the process specified by process-name, the process specified in the /ID or /INDEX qualifier, the next process in the system's process list, the system process, or all processes. The VALIDATE PROCESS command performs an implicit SET PROCESS command under certain uses of its qualifiers and parameters, as noted previously. By default, the VALIDATE PROCESS command validates the SDA current process, as defined in *HP OpenVMS System Analysis Tools Manual*.

Currently, the only validation available is to check free pool packets for POOLCHECK-style corruption. The command is useful only if pool checking is enabled using either the POOLCHECK or the SYSTEM_CHECK system parameters. (For information on these system parameters, see the *HP OpenVMS System Management Utilities Reference Manual*.

If a process is specified using *process-name*, /ADDRESS, /ID, /INDEX, /NEXT, or /SYSTEM, that process becomes the SDA current process for future commands.

CLUE REGISTER

Displays the active register set for the crash CPU. The CLUE REGISTER command is valid only for analyzing crash dumps.

Format

```
CLUE REGISTER   [/CPU [cpu-id | ALL]
                | /PROCESS [/ADDRESS=n | INDEX=n
                | /IDENTIFICATION=n | process-name | ALL]]
```

Parameters

ALL
When used with /CPU, it requests information about all CPUs in the system. When used with /PROCESS, it requests information about all processes that exist in the system.

cpu-id
When used with /CPU, it gives the number of the CPU for which information is to be displayed. Use of the *cpu-id* parameter causes the CLUE REGISTER command to perform an implicit SET CPU command, making the indicated CPU the current CPU for subsequent SDA commands.

process-name
When used with /PROCESS, it gives the name of the process for which information is to be displayed. Use of the *process-name* parameter, the /ADDRESS qualifier, the /INDEX qualifier, or the /IDENTIFICATION qualifier causes the CLUE REGISTER command to perform an implicit SET PROCESS command, making the indicated process the current process for subsequent SDA commands. You can determine the names of the processes in the system by issuing the SHOW SUMMARY command.

The *process-name* can contain up to 15 letters and numerals, including the underscore (_) and dollar sign ($). If it contains any other characters, you must enclose the *process-name* in quotation marks (" ").

Qualifiers

/ADDRESS=*n*
Specifies the PCB address of the desired process when used with CLUE REGISTER/PROCESS.

/CPU [cpu-id | ALL]
Indicates that the registers for a CPU are required. Specify the CPU by its number or use ALL to indicate all CPUs.

/IDENTIFICATION=*n*
Specifies the identification of the desired process when used with CLUE REGISTER/PROCESS.

/INDEX=*n*
Specifies the index of the desired process when used with CLUE REGISTER/PROCESS.

/PROCESS [process-name | ALL]
Indicates that the registers for a process are required. The process should be specified with either one of the qualifiers /ADDRESS, /IDENTIFICATION, or /INDEX, or by its name, or by using ALL to indicate all processes.

Description

The CLUE REGISTER command displays the active register set of the crash CPU. It also identifies any known data structures, symbolizes any system virtual addresses, interprets the processor status (PS), and attempts to interpret R0 as a condition code.

If neither /CPU nor /PROCESS is specified, the parameter (*cpu-id* or *process-name*) is ignored and the registers for the SDA current process are displayed.

CLUE SCSI

Displays information related to SCSI and Fibre Channel.

Format

CLUE SCSI {/CONNECTION=*n* | /PORT=*n* | /REQUEST=*n* | /SUMMARY}

Qualifiers

/CONNECTION=*scdt-address*
Displays information about SCSI connections and decodes the SCSI connection descriptor data structure identified by the SCDT address.

/PORT=*spdt-address*
Displays all or a specific port descriptor identified by its SPDT address.

/REQUEST=*scdrp-address*
Displays information about SCSI requests and decodes the SCSI class driver request packet identified by the SCDRP address.

/SUMMARY
Displays a summary of all SCSI and FC ports and devices and their type and revisions.

SDA$CBB_BOOLEAN_OPER

Performs a Boolean operation on a pair of CBBs.

Format

int sda$cbb_boolean_oper (CBB_PQ input_cbb, CBB_PQ output_cbb, int operation);

Arguments

input_cbb

OpenVMS usage	address
type	CBB structure
access	read only
mechanism	by reference

The address of the first (input) CBB structure.

output_cbb

OpenVMS usage	address
type	CBB structure
access	read/write
mechanism	by reference

The address of the second (output) CBB structure.

operation

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The desired operation from the following list:

- CBB$C_OR—The logical sum of the two CBBs is performed and the result (B = A | B) is written to the output CBB.
- CBB$C_BIC—The logical product with complement of the two CBBs is performed and the result (B = B & ~A) is written to the output CBB.

Description

The desired Boolean operation is performed on the two CBB structures, and the result is written to the second (output) structure.

Condition Values Returned

SS$_WASCLR	The value 0 is returned if no bits are set in the resulting output CBB.
SS$_WASSET	The value 1 is returned if any bit is set in the resulting output CBB.
SS$_BADPARAM	The number of valid bits in the input and output CBBs are different.

SDA$CBB_CLEAR_BIT

Clears the specified bit in a CBB.

Format

```
int sda$cbb_clear_bit   (CBB_PQ cbb, int bit);
```

Arguments

cbb

OpenVMS usage	address
type	CBB structure
access	read/write
mechanism	by reference

The address of the CBB structure to be modified.

bit

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The bit in the CBB to be cleared. If the bit number is -1, clears all bits.

Description

The specified bit (or all bits) in the CBB is cleared.

Condition Values Returned

SS$_NORMAL	Successful completion
SS$_BADPARAM	The bit number is out of range

SDA$CBB_COPY

Copies the contents of one CBB to another.

Format

```
int sda$cbb_copy   (CBB_PQ input_cbb, CBB_PQ output_cbb);
```

Arguments

input_cbb

OpenVMS usage	address
type	CBB structure
access	read only
mechanism	by reference

The address of the CBB structure to be copied.

output_cbb

OpenVMS usage	address
type	CBB structure
access	write only
mechanism	by reference

The address of the CBB structure to receive the copy.

Description

The specified CBB is copied.

Condition Values Returned

None

SDA$CBB_FFC

Locates the first clear bit in a CBB.

Format

int sda$cbb_ffc (CBB_PQ cbb, int start_bit);

Arguments

cbb

OpenVMS usage	address
type	CBB structure
access	read only
mechanism	by reference

The address of the CBB structure to be searched.

start_bit

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The first bit in the CBB to be checked.

Description

The CBB structure is searched, starting at the specified bit, for a clear bit.

Condition Values Returned

Bit_number	If a clear bit is found, its bit number is returned. If no clear bit is found (all bits from start_bit to cbb->cbb$l_valid_bits are set), then the number of valid bits is returned.

SDA$CBB_FFS

Locates the first set bit in a CBB.

Format

```
int sda$cbb_ffs    (CBB_PQ cbb, int start_bit);
```

Arguments

cbb

OpenVMS usage	address
type	CBB structure
access	read only
mechanism	by reference

The address of the CBB structure to be searched.

start_bit

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The first bit in the CBB to be checked.

Description

The CBB structure is searched, starting at the specified bit, for a set bit.

Condition Values Returned

Bit_number	If a set bit is found, its bit number is returned. If no set bit is found (all bits from start_bit to cbb->cbb$l_valid_bits are clear), then the number of valid bits is returned.

SDA$CBB_INIT

Initializes a CBB structure to a known state.

Format

```
void sda$cbb_init   (CBB_PQ cbb);
```

Arguments

cbb

OpenVMS usage	address
type	CBB structure
access	read only
mechanism	by reference

The address of the CBB structure to be initialized.

Description

The fields of the CBB that describe its layout are initialized as necessary for a CPU CBB. The actual bitmask is zeroed.

Condition Values Returned

None

SDA$CBB_SET_BIT

Sets the specified bit in a CBB.

Format

int sda$cbb_set_bit (CBB_PQ cbb,int bit);

Arguments

cbb

OpenVMS usage	address
type	CBB structure
access	read/write
mechanism	by reference

The address of the CBB structure to be modified.

bit

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The bit in the CBB to be set. If the bit number is -1, set all bits.

Description

The specified bit (or all bits) in the CBB is set.

Condition Values Returned

SS$_NORMAL	Successful completion.
SS$_BADPARAM	The bit number is out of range.

SDA$CBB_TEST_BIT

Tests the specified bit in a CBB.

Format

```
int sda$cbb_test_bit   (CBB_PQ cbb,int bit);
```

Arguments

cbb

OpenVMS usage	address
type	CBB structure
access	read only
mechanism	by reference

The address of the CBB structure to be tested.

bit

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The bit in the CBB to be tested.

Description

The specified bit in the CBB is tested and its value returned.

Condition Values Returned

SS$_WASSET	The specified bit was set.
SS$_WASCLR	The specified bit was clear.
SS$_BADPARAM	The bit number is out of range.

SDA$DELETE_PREFIX

Deletes all symbols with the specified prefix.

Format

```
void sda$delete_prefix    (char *prefix);
```

Arguments

prefix

OpenVMS usage	char_string
type	character string
access	read only
mechanism	by reference

The address of the prefix string.

Description

This routine searches the SDA symbol table and deletes all symbols that begin with the specified string.

Condition Values Returned

None

SDA$FID_TO_NAME

Translates a file identification (FID) into the equivalent file name.

Format

int sda$fid_to_name (char *devptr, unsigned short *fidptr, char *bufptr, int buflen);

Arguments

devptr

OpenVMS usage	char_string
type	character string
access	read only
mechanism	by reference

The address of the device name string. The device name must be supplied in allocation-class device name (ALLDEVNAM) format, but any leading underscores or trailing colons are ignored.

fidptr

OpenVMS usage	address
type	file identification
access	read only
mechanism	by reference

The address of the three-word file identification.

bufptr

OpenVMS usage	char_string
type	character string
access	write only
mechanism	by reference

The address of a string buffer into which to store the file name string.

buflen

OpenVMS usage	longword
type	longword (unsigned)
access	read only
mechanism	by value

The maximum length of the string buffer.

Description

When analyzing the current system, this routine calls LIB$FID_TO_NAME to translate the file identification into a file name. When analyzing a dump, if there is a file data collection available and the specified disk and file identification is included in the collection, the recorded file name is returned. If there is no collection (for the entire system, this disk, or just this file), this routine returns the error condition SDA$_NOCOLLECT.

Condition Values Returned

SDA$_SUCCESS	File identification successfully translated.
SDA$_COLLECT	No collection available for the system, the specified disk, or the file identification.
Others	An error occurred when LIB$FID_TO_NAME was called.

SDA$GET_FLAGS

Obtains environment flags that indicate how SDA is being used.

Format

```
int sda$get_flags   (SDA_FLAGS *flagaddr);
```

Arguments

flagaddr

OpenVMS usage	address
type	SDA_FLAGS structure
access	write only
mechanism	by reference

The address of the location where the environment flags are to be returned.

Description

SDA provides a set of flag bits that indicate whether it is being used to analyze the current system, a system dump, a process dump, and so on. The set of bits is defined in SDA_FLAGSDEF.H in SYS$LIBRARY:SYS$LIB_C.TLB.

Condition Values Returned

None

3.16.3 ANALYZE Command Qualifier

The new SDA ANALYZE command /COLLECTION qualifier indicates to SDA that the file ID translation data or unwind data can be found in a separate file. If you specify this qualifier, it should follow the /CRASH_DUMP qualifier in the command string. Use the following format:

```
/CRASH_DUMP/COLLECTION = collection-file-name
```

SDA can provide additional information when analyzing a dump if a collection has been made of file identification translation data (on both OpenVMS Alpha and OpenVMS for Integrity servers) and of unwind data (on OpenVMS for Integrity servers only). This data is usually saved when the dump file is copied using the SDA COPY/COLLECT command, but it can be saved to a separate file using the COLLECT/SAVE command.

By default, COLLECT/SAVE creates a .COLLECT file with the same name and in the same directory as the dump file. A subsequent ANALYZE/CRASH_DUMP command uses this file automatically. If the collection file is in a different location or if the collection previously appended to the dump file is incomplete (for example, a disk was not mounted at the time of the SDA COPY), the /COLLECTION qualifier can be used to specify an alternate collection file.

At least one field of the collection file name must be specified, and other fields default to the highest generation of the same file name and location as the dump file, with a file type of .COLLECT.

3.16.4 DUMP Command Qualifiers

The SDA DUMP command has the following new qualifiers:

- /BYTE—Outputs each data item as a byte.
- /NOSUPPRESS—Indicates that SDA should not suppress leading zeroes when displaying data in hexadecimal format.
- /WORD—Outputs each data item as a WORD.

3.16.5 SEARCH Command Qualifier

The SDA SEARCH command has the new /IGNORE_CASE qualifier, which indicates to SDA that, when searching for a string, the case of any alphabetic characters should be ignored. The default behavior is to search for an exact match. This qualifier is ignored for value searches.

3.16.6 New SHOW CLUSTER Command Qualifier

The SDA SHOW CLUSTER command has the new /CIRCUIT=*pb-addr* qualifier, which displays only the OpenVMS Cluster system information for a specific path, where *pb-addr* is the address of its path block. This qualifier is mutually exclusive with the /ADDRESS=*n*, /CSID=*csid*, and /NODE=*name* qualifiers. If you specify the /CIRCUIT=*pb-addr* qualifier, the SHOW CLUSTER command displays only the information from the specified path block.

3.16.7 SHOW CRASH Qualifier

The SDA SHOW CRASH command has the new /ALL qualifier, which displays exception data for all CPUs. By default, the registers (on Alpha) or exception frame contents (on Integrity servers) are omitted from the display for any CPUs with CPUEXIT or DBGCPUEXIT bugchecks.

3.16.8 SHOW DUMP Command Qualifiers

The following new qualifiers have been added to the SHOW DUMP command:

- /COLLECTION[= { ALL | *n* }]

  Displays the contents of the file identification and/or unwind data collection appended to a copy of the dump using COPY/COLLECT or written to a separate collection file using COLLECT/SAVE. By default, a summary of the collection is displayed. You can specify that the details of a single entry or all entries are to be displayed. The *n* is the start block number of the collection entry, as displayed in the collection summary.

- /FILE= { COLLECTION | DUMP }

  If a separate collection file is in use, the /FILE qualifier indicates to which file the SHOW DUMP command applies. By default, SHOW DUMP/SUMMARY, SHOW DUMP/HEADER, SHOW DUMP/COLLECTION and SHOW DUMP/ALL commands apply to both files. By default, SHOW DUMP/BLOCK applies to the dump file. All other qualifiers can apply only to the dump file.

3.16.9 SDA SHOW PROCESS Qualifier

The SDA SHOW PROCESS command has the new /CHECK qualifier, which checks all free process pool packets for POOLCHECK-style corruption,in exactly the same way that the system does when generating a POOLCHECK crash dump.

3.16.10 Keywords Added to SHOW RESOURCES/STATUS Command

The following new keywords have been added to the SHOW RESOURCES/STATUS qualifier:

- RM_FORCE—Forced tree move
- RM_FREEZE—Freeze resource tree on this node
- RM_INTEREST—Remaster due to master having no interest
- XVAL_VALID—Last value block was long block

3.16.11 SHOW UNWIND Qualifier

The SDA SHOW UNWIND command has the new qualifier, /IMAGE=*name*, which displays the details of every unwind descriptor for the specified system images (wildcards allowed).

3.17 System Parameters

A number of system parameters are introduced in OpenVMS Version 8.3. The following table contains brief descriptions of these new parameters. (More detailed descriptions of the parameters are in the *HP OpenVMS System Management Utilities Reference Manual*.)

Parameter	Description
EXECSTACKPAGES	(Alpha and I64) EXECSTACKPAGES controls the number of pages allocated for each RMS exec stack.

Parameter	Description
GB_CACHEALLMAX	(Alpha and I64) If a file is connected to RMS with the RMS global buffer DEFAULT option enabled, the number of blocks cached is either a maximum of the GB_CACHEALLMAX parameter or a percentage of the file, whichever results in a larger global count.
GB_DEFPERCENT	(Alpha and I64) If a file is connected to RMS with the RMS global buffer "DEFAULT" option enabled, either a percentage (GB_DEFPERCENT) of the file is cached or up to GB_CACHEALLMAX blocks of it are cached, whichever results in a larger global buffer count.
IO_PRCPU_BITMAP	(Alpha and I64) This parameter forms a bitmap representing up to 1024 CPUs. Bits set in this bitmap indicate CPUs that are available for use as Fast Path preferred CPUs. IO_PRCPU_BITMAP defaults to all bits set. (CPU 0 through CPU 1023 are all enabled for Fast Path port assignment.) You may want to disable the primary CPU from serving as a preferred CPU by leaving its bit clear in IO_PRCPU_BITMAP. This reserves the primary CPU for non-Fast-Path IO operations to use.
LOCKRMWT	Can have a value from 0 to 10 and defaults to 5. Remaster decisions are based on the difference in lock remaster weights between the master and a remote node. LOCKRMWT is a dynamic parameter.
SCD_HARD_OFFLD	The scheduler hard off-load parameter is a CPU bitmask parameter. The bits correspond to CPU IDs. For any bit set, the OpenVMS scheduler does not schedule processes on this CPU unless the process has hard affinity set for the CPU. The bit corresponding to the primary CPU is ignored. SCH_HARD_OFFLD is a DYNAMIC parameter.
SCH_SOFT_OFFLD	The scheduler soft off-load parameter is a CPU bitmask parameter. The bits correspond to CPU IDs. For any bit set, the OpenVMS scheduler tries to avoid scheduling processes on this CPU. However, if no other idle CPUs exist, processes are still scheduled on this CPU. SCH_SOFT_OFFLD is a DYNAMIC parameter.
SCHED_FLAG	This special parameter is used by HP and is subject to change. Do not change this parameter unless HP recommends that you do so.
SMP_CPU_BITMAP	(Alpha and I64) This parameter indicates that the corresponding CPU is a bitmap representing up to 1024 CPUs. Each bit set in this bitmap indicates that the corresponding CPU automatically attempts to join the active set in an OpenVMS symmetric multiprocessing environment when the instance is booted.
VCC_PAGESIZE	(Alpha and I64) VCC_PAGESIZE is a special parameter reserved for HP use only. Extended File Cache intends to use this parameter in future versions.
VCC_RSVD	(Alpha and I64) VCC_RSVD is a special parameter reserved for HP use only. Extended File Cache intends to use this parameter in future versions.

3.18 System Service Logging Enhancements

The system service logging (SSLOG) mechanism has been enhanced for OpenVMS Version 8.3:

- When a system service request is logged, the CPU, kernel thread, and POSIX threads IDs from which the service was requested are now recorded.

  The ANALYZE/SSLOG utility displays this new information with other details from each entry.

  You can selectively display entries based on these characteristics through new /SELECT values:

Value	Description
CPU	CPU ID
KTID	Kernel thread ID
TID	POSIX thread ID

- It is possible to have system service requests logged only to the process's buffers and not to a file. This is, however, of very limited use and not recommended, because accessing any of the information logged would require finding the logging buffers in memory or in a crashdump and formatting them manually.

  You specify this type of logging through a new value for the SET PROCESS/SSLOG qualifier FLAGS value:

  ```
  SET PROCESS/SSLOG=(STATE=ON[, FLAGS=[NO]FILE])
  ```

  The default value for this flags is FILE.

- A process can be created with system service logging enabled. This happens automatically when a process with logging enabled creates a subprocess: logging characteristics of the parent are propagated to the child.

  Also, you can explicitly create a process with logging enabled with a new $RUN command qualifier. The syntax for the RUN command is

  ```
  RUN /SSLOG_ENABLE=([COUNT=x][,SIZE=y]
     [,FLAGS=([NO]ARG,[NO]FILE))]
  ```

  Alternatively, you can request the $CREPRC system service with the following new parameters:

  – Flag PRC$M_SSLOG_ENABLE in argument **stsflag**, when set, requests that system service logging be enabled in the new process.

  – You can specify logging characteristics through item list entry types PRCC_SSLOG_FLAGS, PRCC_SSLOG_BUFSIZE, and PRC$C_SSLOG_BUFCNT.

  Regardless of how the process is created, logging does not begin until after the process's first image has been fully activated.

System service logging is described in detail in the *HP OpenVMS System Analysis Tools Manual*.

3.19 SYS$ACM-Enabled LOGINOUT.EXE and SETP0.EXE Images for LDAP Authentication

Important

The images described in this section are "pre-production" images and are not qualified for production use. Once additional rigorous "production-quality" testing and qualification is completed, a maintenance update (ECO) will be made available to allow for production use deployments of the SYS$ACM-enabled **loginout** and **setp0** images.

This release provides optional LOGINOUT.EXE and SETP0.EXE (SET PASSWORD) images that use the SYS$ACM system service for user authentication and password changes.

When these images are used, login and password change requests are sent to the SYS$ACM service and handled by the ACME_SERVER process's authentication agents.

A VMS authentication agent is configured by default to service standard VMS login and password-change requests. In addition, you can install an LDAP authentication agent that services login and password-change requests using an LDAP version 3 directory server.

For more information, see the SYS$HELP:ACME_DEV_README.TXT file.

3.20 Time Zones Added

OpenVMS Version 8.3 provides 544 time zones based on the time-zone public database named tzdata2006b. Five new time zones have been added in OpenVMS Version 8.3:

Australia/Currie
America/Coral_Harbour
America/Indiana/Vincennes
America/Indiana/Petersburg
America/Moncton

An additional 12 time zones were added in Version 8.2–1 but were not documented:

America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/Comodrivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/San_Juan
America/Argentina/Tucuman
America/Argentina/Ushuaia
Europe/Mariehamn

These new time zones will be added to an appendix in the *HP OpenVMS System Manager's Manual* the next time it is updated.

The following time zones have been deleted:

- SystemV/AST4ADT
- SystemV/EST5EDT
- SystemV/CST6CDT
- SystemV/MST7MDT
- SystemV/PST8PDT
- SystemV/YST9YDT
- SystemV/AST4
- SystemV/EST5
- SystemV/CST6
- SystemV/MST7
- SystemV/PST8
- SystemV/YST9
- SystemV/HST10

Note

With the passage of the Energy Policy Act in 2005 in the United States, starting in March 2007; daylight saving time (DST) will begin on the second Sunday in March (instead of the current first Sunday in April). DST will end on the first Sunday in November (instead of the current last Sunday in October). The latest time-zone rules have been incorporated into OpenVMS Version 8.3.

Patch kits for OpenVMS Versions 7.3–2, 8.2, and 8.2–1 are provided on the OpenVMS Alpha Version 8.3 Operating System CD and the OpenVMS for Integrity servers Version 8.3 DVD.

3.21 Virtual LAN (VLAN) Support in OpenVMS

Virtual LAN (VLAN) is a mechanism for segmenting a LAN broadcast domain into smaller sections. The IEEE 802.1Q specification defines the operation and behavior of a VLAN. The OpenVMS implementation adds IEEE 802.1Q support to selected OpenVMS LAN drivers so that OpenVMS can now route VLAN tagged packets to LAN applications using a single LAN adapter.

You can use VLAN to do the following:

- Segment specific LAN traffic on a network for the purposes of network security or traffic containment, or both.
- Use VLAN isolated networks to simplify address management

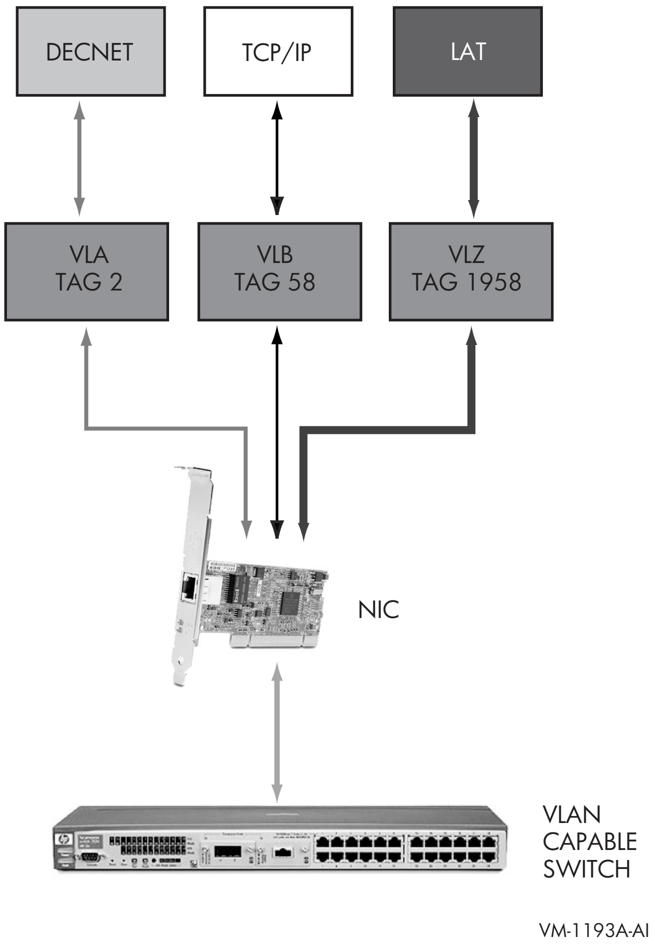
VLAN Design

In OpenVMS, VLAN presents a virtual LAN device to LAN applications. The virtual LAN device associates a single IEE 802.1Q tag with communications over a physical LAN device. The virtual device provides the ability to run any LAN application (for example, SCA, DECnet, TCP/IP, or LAT) over a physical LAN device, allowing host-to-host communications as shown in Figure 3–1.

Note

DECnet-Plus and DECnet Phase IV can be configured to run over a VLAN device.

Figure 3–1 Virtual LAN



OpenVMS VLAN has been implemented through a new driver, SYS$VLANDRIVER.EXE, which provides the virtual LAN devices. Also, existing LAN drivers have been updated to handle VLAN tags. LANCP.EXE and LANACP.EXE have been updated with the ability to create and deactivate VLAN devices and to display status and configuration information.

The OpenVMS VLAN subsystem was designed with particular attention to performance. Thus, the performance cost of using VLAN support is negligible.

When configuring VLAN devices, keep in mind that VLAN devices share the same locking mechanism as the physical LAN device. For example, running OpenVMS cluster protocol on a VLAN device along with the underlying physical LAN device does not result in increased benefit and might, in fact, hinder performance.

3.21.1 VLAN Support Details

All supported Gigabit and 10-Gb (I64-only) LAN devices are capable of handling VLAN traffic on Alpha and I64 systems.

The following list describes additional details of VLAN-related support:

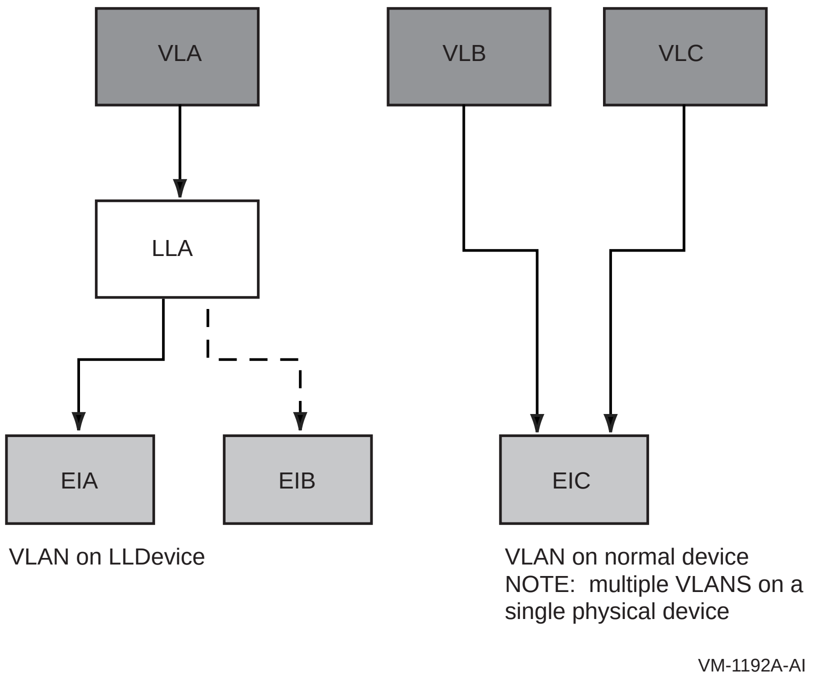
- Switch support

  For VLAN configuration, the only requirement of a switch is conformance to the IEEE 802.1Q specification. The VLAN user interface to the switch is not standard; therefore, you must pay special attention when you configure a switch and especially when you configure VLANs across different switches.

- LAN Failover support

  Figure 3–2 illustrates LAN Failover support.

Figure 3–2 LAN Failover Support



  You can create VLAN devices using a LAN Failover set as a source if all members of the set are VLAN-capable devices. However, you cannot build a Failover set using VLAN devices.

- Supported capabilities

  VLAN devices inherit the capability of the underlying physical LAN device, including fast path, auto-negotiation, and jumbo frame setting. If a capability needs to be modified, you must modify the underlying physical LAN device.

- Restrictions

  No support exists for satellite booting over a VLAN device. The OpenVMS LAN boot drivers do not include VLAN support; therefore, you cannot use a VLAN device to boot an OpenVMS system.

  Currently, no support exists in OpenVMS for automatic configuration of VLAN devices. You must create VLAN devices explicitly using LANCP commands.

3.21.2 Managing VLAN on Your System

Before creating a VLAN device, make sure that the hosting VLAN-capable physical LAN device is connected to a VLAN-capable switch. Also make sure that the selected switch port is configured to handle VLAN-tagged traffic.

The following sections contain additional VLAN management details.

3.21.2.1 Probing a Switch Port

To make it easier to manage VLAN devices, OpenVMS LAN includes limited support for IEEE 802.1Q management functions. A LANCP qualifier helps you probe a switch port and list VLAN configuration information. The new command is the following:

```
LANCP> SHOW DEVICE PHYSICAL-LAN-DEVICE/VLAN
```

After you enter the command, LANCP listens for IEEE 802.1Q GVRP (Generic Attribute Registration Protocol) VLAN Registration Protocol packets and displays the following:

- The VLAN tags that have been configured on the switch port
- The VLAN devices that have been configured on the physical LAN device

For example:

```
LANCP> SHOW DEVICE LLB /VLAN

Listening for VLAN configuration on LLB0 ......
   VLAN tag 190 configured as VLB
   VLAN tag 206 configured as VLJ
   VLAN tag 207 not configured
```

This command shows VLAN information only if GVRP capability is enabled on the switch port.

3.21.2.2 Creating a VLAN Device

To create a VLAN device, enter a LANCP command using the following format:

```
LANCP> SET DEVICE VLc/VLAN_DEVICE=PHYSICAL-LAN-DEVICE/
TAG=value
```

where:

- VL*c* is the name of the virtual LAN device (*c* is the controller letter a - z).
- PHYSICAL-LAN-DEVICE is the LN device that will host the VLAN.
- *value* is the IEEE 802.1Q tag. (The valid range is 1 - 4095.)

For example:

```
LANCP> SET DEVICE VLA/VLAN=EIB/TAG=42
```

This command fails if the physical LAN device does not exist, if the physical LAN device is not VLAN-capable, or if the VLAN tag is invalid.

Associating a Text Description with a LAN Device

Also new in this version of OpenVMS, you can associate a text description with a LAN device. You do this by entering a LANCP SET or DEFINE DEVICE command with the qualifier /DESCRIPTION=<quoted-string> to provide the additional context. For example, to identify a VLAN device as part of the “Finance VLAN”, enter the following command:

```
LANCP> SET DEVICE VLA/DESCRIPTION="Finance VLAN"
```

3.21.2.3 Deactivating a Virtual LAN Device

Note

The deactivation functionality has not yet been completed at the time of Field Test. Watch for updates during Field Test for this capability.

To deactivate a VLAN device, use the following command format:

```
LANCP> SET DEVICE VLc/NOVLAN
```

This command fails if the device is in use, that is, if other applications are still using the device.

3.21.2.4 Displaying VLAN Device Information

To display information about the VLAN device, enter the LANCP commands SHOW DEVICE and SHOW CONFIGURATION. For example:

```
LANCP> SHOW DEVICE VLK/CHARACTERISTICS

Device Characteristics VLK0:
                    Value   Characteristic
                    _____   ______________
                    ...
                    "206"   VLAN 802.1Q tag
                      "1"   VLAN device flags
     "Procurve 2315 P15"   VLAN description
                  Link Up   Link state

Device Parent Medium/User Version Link Speed Duplex Size     MAC Address    Current Address   Type
------ ------ ----------- ------- ---- ----- ------ ---- ----------------- ----------------- ----
EWA0          Ethernet    X-51    Up   1000  Full   1500 00-D0-59-61-72-F3 AA-00-04-00-1B-4D UTP DEGXA-TA
EWB0          Ethernet    X-51    Up    100  Full   1500 00-D0-59-61-72-D8 00-D0-59-61-72-D8 UTP DEGXA-TA
EWC0          Ethernet    X-59    Up   1000  Full   1500 00-60-CF-21-71-9C AA-00-00-21-71-9C UTP DEGPA-TA
EWD0          Ethernet    X-59    Up   1000  Full   1500 00-60-CF-20-9A-C6 00-60-CF-20-9A-C6 UTP DEGPA-TA
EIA0          Ethernet    X-16    Up   1000  Full   1500 00-12-79-9E-20-AE AA-00-04-00-1B-4D UTP AB352A
EIB0          Ethernet    X-16    Up   1000  Full   1500 00-12-79-9E-20-AF 00-12-79-9E-20-AF UTP AB352A
LLB0          Ethernet    X-19    Up   1000  Full   1500 AA-00-00-21-71-9C AA-00-00-21-71-9C DEGPA-TA
VLB0          Ethernet    X-BA1   Up   1000  Full   1500 AA-00-00-21-71-9C AA-00-00-21-71-9C LLB
VLC0          Ethernet    X-BA1   Up   1000  Full   1500 00-12-79-9E-20-AF 00-12-79-9E-20-AF UTP EIB
VLD0          Ethernet    X-BA1   Down  100  Full   1500 00-00-00-00-00-00 00-00-00-00-00-00
VLJ0          Ethernet    X-BA1   Up   1000  Full   1500 AA-00-00-21-71-9C AA-00-00-21-71-9C LLB
VLK0          Ethernet    X-BA1   Up   1000  Full   1500 00-12-79-9E-20-AE AA-00-04-00-1B-4D UTP EIA
```

3.21.3 VLAN Troubleshooting

Most VLAN problems are related to configuration. A list of things to check when you are troubleshooting a VLAN problem:

1. To OpenVMS, not all LAN devices are VLAN-capable. If you attempt to create a VLAN device on a non-VLAN-capable device, LANCP displays an error message.

To verify that a LAN device is VLAN-capable, use SDA to check device characteristics by entering the following commands:

```
$ ANALYZE/SYSTEM

SDA> SHOW LAN/DEVICE=physical-device-name
or
SDA> LAN DEVICE/DEVICE=physical-device-name
```

VLAN bit 4 should be set in the device characteristics, which the text string "VLAN" indicates.

2. Verify that VLAN capability is enabled on the switch port that is connected to your LAN device and that the correct VLAN tag is configured. If GVRP is enabled on the switch, you can verify that the VLAN tag is enabled by entering the following LANCP command:

```
LANCP> SHOW DEVICE physical-device-name/VLAN
```

This command displays the VLAN tags configured on the switch port. Next, verify that the tag displayed is the one that was used to create the VLAN device.

3. Verify that the VLAN device was configured correctly. Enter the following command to see the characteristics and status maintained by the VLAN driver:

```
LANCP> SHOW DEVICE vlan-device-name/INTERNAL_COUNTERS
```

For example:

```
LANCP> SHOW DEVICE VLC/INTERNAL_COUNTERS

Device Internal Counters VLC0:
                  Value  Counter
                  _____  _______
                         --- Internal Driver Counters ---
             "     EIB"  Device name
               00000001  Device Flag 1 <online>
                    190  VLAN Tag ID
               86514000  Physical LSB
                  11834  Failure status
      FFFFFFFF 805E28CC  Failure PC
```

Check the following:

a. The device name and tag should be the same as those specified when you created the VLAN device.

b. Verify that the "online" bit is set on the Device Flag 1 field; if not, the failure status might provide more information.

c. The physical LSB field is the address of the LAN physical device LSB (LAN Station Block) structure. To look at the characteristics and status of this device, enter the following commands:

```
$ ANALYZE/SYSTEM

SDA> LAN DEVICE/ADDRESS=physical LSB address
```

For more information about OpenVMS VLAN support, see the *HP OpenVMS System Management Utilities Reference Manual*.

3.22 Volume Shadowing for OpenVMS

The following new features for HP Volume Shadowing for OpenVMS are available in OpenVMS Version 8.3:

- Automatic bitmap creation on volume processing
- New SET SHADOW qualifier, /RESET

3.22.1 Automatic Bitmap Creation on Volume Processing

Automatic bitmap creation on volume processing means that an existing HBMM bitmap is made available to function as a minicopy bitmap when connectivity to one or more shadow set members is lost and is not restored during the shadow member timeout period.

When such connectivity is lost, the shadow set is paused for volume processing—that is, writes and reads are temporarily suspended until connectivity is restored or the timeout period (established by the value of SHADOW_MBR_TMO) expires, whichever comes first.

If connectivity is not restored by the end of the timeout period, the member or members are expelled from the shadow set, read and write I/O to the remaining member or members resumes, and the bitmap keeps track of the writes. The bitmap, whose name has changed from HBMM*x* to rrse*x*, functions as a minicopy bitmap for the member or members that were expelled.

Note

While one or two members are expelled and after all members are restored to membership in the shadow set, the HBMM bitmap functionality remains in effect. The HBMM bitmap functionality is useful in the case of an expelled member only when the shadow set has three members and one member is expelled.

When connectivity is restored to one of the expelled shadow set members, you can mount it back into the shadow set. If the expelled member's metadata matches a bitmap that exists, it is used for a minicopy operation to restore that member to the shadow set. If a second shadow set member was removed at the same time, that member can also use that bitmap. After the members are restored to the shadow set, the name of the bitmap reverts to its HBMM bitmap name.

The reasons to minimize the time when one or more members are expelled from a shadow set are:

- During a period of reduced membership of the shadow set, data availability is at risk.
- If a shadow set member is expelled, reads and writes to the remaining members continue. The more writes that take place before the expelled member or members are returned, the longer it takes to restore the member or members to the shadow set. This is especially significant in a disaster tolerant (DT) configuration.

Before the introduction of automatic bitmap creation on volume processing, returning expelled members to a shadow set, after connectivity was restored, was a lengthy process. The expelled members could be returned only by undergoing a full copy. The availability of a bitmap enables the use of a minicopy operation, which takes considerably less time than a full copy operation.

To enable automatic bitmap creation on volume processing, you need to establish an HBMM policy for the shadow sets, and include the new MULTIUSE keyword in the policy. For more information, refer to the HBMM chapter in the *HP OpenVMS Version 8.2 New Features and Documentation Overview* manual.

3.22.2 New SET SHADOW /RESET Qualifier

The /RESET qualifier to the SET SHADOW command is introduced in this release. SET SHADOW/RESET=COUNTERS resets the shadowing-specific counters that are maintained for each shadow set.

The counters that are reset to 0 are:

HBMM Reset Count
Copy Hotblocks
Copy Collisions
SCP Merge Repair Cnt
APP Merge Repair Cnt

You can display the current settings of these counters using the SHOW SHADOW command.

The HBMM Reset Count refers to how many times the RESET_THRESHOLD value was met. The RESET_THRESHOLD is the setting which determines how frequently a bitmap is cleared. With the ability to clear the HBMM Reset Count, system managers can better gauge the rate of threshold resets.

For a complete description of SET SHADOW/RESET, refer to the *HP OpenVMS DCL Dictionary: N–Z* and DCL Help.

4 Mastering Optical Media on OpenVMS

This chapter describes the creation (or mastering) of CD or DVD media on OpenVMS.

The process of mastering CD or DVD media includes the following tasks:

1. Creating a disk volume structure in a staging area
2. Populating that structure with the required files
3. Copying the master onto the target optical media

On OpenVMS, you must use a logical disk (LD) device as the staging area and DCL commands such as INITIALIZE, MOUNT, COPY, and BACKUP to generate and populate the disk volume in the staging area. You can then copy the contents of the disk volume by using the COPY/RECORDABLE_MEDIA command.

4.1 LD, CD, and DVD Device Concepts

The following sections discuss concepts that pertain to mastering optical media on OpenVMS.

4.1.1 Logical Disk Devices

A logical disk (LD) device provides a mechanism for staging the master copy of the data to be written to the optical media. You can create the source for the recording operation using an LD disk device and then enter the COPY/RECORDABLE_MEDIA command to transfer the master onto the optical media.

You use the LD utility to create and manage LD disk devices. You can then initialize, mount, and access these LD disk devices using standard OpenVMS DCL commands.

For more information about LD disk devices, see the *HP OpenVMS System Manager's Manual*.

4.1.2 CD and DVD Devices

You can use various recording formats with optical media devices. In general, OpenVMS can read formats that correspond to the target device you use.

OpenVMS can record the following four media formats:

Format	Description
CD-R	Compact Disc Recordable
CD-RW	Compact Disc Rewritable
DVD+R	Digital Versatile Disc Recordable

Format	Description
DVD+RW	Digital Versatile Disc Rewritable

The particular characteristics and capabilities of the target CD and DVD devices are specific to the system, the recording device, and the recording media. For example, the local hardware and software configuration can further restrict the maximum permissible CD recording speed to a value less than the speed supported by the CD recording device. You might attempt to record a CD from an OpenVMS system that does not have the I/O bandwidth you need to keep the data cache of the target CD device from underflowing. However, such attempts can result in recording errors and failures, and can waste recording media.

Recording devices can support a variety of recording formats and media. Conversely, OpenVMS or a particular device might not support a particular recording format. For the currently supported device hardware and their associated platform configurations, see the following web site:

`http://www.hp.com/go/server/`

Find your particular I64 or Alpha platform, and then look for the support matrix for that platform.

4.2 General Steps for Mastering Data Disks

The steps for mastering (sometimes called recording or burning) optical media are the following:

1. Start to create an OpenVMS logical disk (LD) by entering the following command:

   ```
   $ @SYS$STARTUP:LD$STARTUP.COM
   ```

   Note

   LD$STARTUP requires the TMPMBX, NETMBX, and SYSLCK privileges. The COPY/RECORDABLE_MEDIA command, used later in these steps, is installed with the necessary privileges.

   Enter this command only once each time the OpenVMS system is booted. To have the system perform the command for you, include the command in your site-specific SYS$MANAGER:SYSTARTUP_VMS.COM system startup procedure. In this way, the command executes each time the OpenVMS system is bootstrapped.

2. Create a logical disk (LD) to act as the staging area for your media master. This LD disk device appears and operates like a standard physical disk device but also provides flexibility because it can be easily sized or resized. In addition, you can create or delete the device as needed.

   The LD driver, which enables you to connect to and manage an LD disk device, uses a back-up storage file that allows the contents of the LD disk device to be preserved over a reboot. The capacity of the LD disk device—and the corresponding size of the back-up file—must be equal to or larger than the size of the files and the volume structure data to be stored. The capacity of the LD disk device must also be equal to or smaller than the capacity of the target optical media. The contents of the master must fit on the target media.

Approximate maximum capacities are usually the following:

Media	Maximum Blocks	Capacity
CD-R	1,200,000 blocks	600 MB/s
CD-RW	1,400,000 blocks	700 MB/s
Single-layer DVD+R	9,180,416 blocks	4.6 GB/s
Single-layer DVD+RW	9,180,416 blocks	4.6 GB/s

You can create sizes up to the maximum for the target media. Because optical media uses a sector size of 4 blocks (2048 bytes), you must always create and use an LD disk device with a capacity that is a multiple of 4 blocks. HP recommends that you use a multiple of 16 blocks.

3. To create your LD master, first create an LD backing storage file for the master. Use a command similar to the following:

   ```
   $ LD CREATE /size=9180416 filespec.ISO
   ```

   You need to create this LD storage file only once.

4. Connect the LD storage file to an LD logical disk. Use a command similar to the following:

   ```
   $ LD CONNECT filespec.ISO LDA1:
   ```

   You need to reconnect the LD disk device once each time the OpenVMS system bootstraps. You can include the LD CONNECT command in the SYSTARTUP_VMS.COM site-specific system startup and have the system execute the command for you each time the system bootstraps.

5. Prepare the master for use.

   Consider erasing the LD master completely before proceeding. This action prevents you from unintentionally disclosing confidential information about your local system. You can erase the disk master in various ways, including using the DCL command INITIALIZE/ERASE if you are creating an ODS-2 or ODS-5 volume structure.

   If you choose to use the OpenVMS ODS-2 or ODS-5 volume structure for your target media, use the DCL command INITIALIZE to create the volume structures. Then use the standard MOUNT command to make the master disk volume accessible to other OpenVMS commands:

   Use commands similar to the following:

   ```
       $ INITIALIZE LDA1: volume-label -
           /SYSTEM [/ERASE] [/...] -
           [/CLUSTER=n] [/STRUCTURE=n] [/...]

       $ MOUNT LDA1: volume-label
   ```

6. Once the volume structure is available, you can copy the data onto the master.

   The data to be copied onto the LD master can include data files, installation kits, executable images, tools, or other files. As with a standard physical disk formatted as an ODS-2 or ODS-5 volume, you can use the BACKUP, COPY, CREATE/DIRECTORY, and other standard DCL commands and procedures to create the contents of the LD master.

If you plan to use ODS-2 or ODS-5 volume structures, avoid placing OpenVMS security identifiers or ACLs on the master. These are system specific and can unexpectedly allow or deny access when you mount or access the recorded media on other OpenVMS systems.

7. After copying your selected contents onto the LD disk device containing the master, dismount the device using a command similar to the following:

```
$ DISMOUNT LDA1:
```

8. Record the contents of the LD master onto the optical media.

   First place the appropriate blank media in the optical media disk drive. Then enter a command similar to the following:

```
   $ COPY/RECORDABLE_MEDIA LDA1: DQA0: -
   _$ [/FORMAT][/BELL][/SPEED=speed][/VERIFY]
```

   This command copies the contents of the LDA1: master to the target device.

   In this example, note the following:

   - The target device is assumed to be DQA0: and is assumed to have rewritable media loaded. The particular target device name can vary according to your local hardware configuration.
   - The /FORMAT qualifier is applicable only with rewritable media; it causes the rewritable media to be erased and to be prepared for recording.
   - Specifying the /SPEED qualifier reduces the recording speed from the default speed calculation; this might be necessary if your attempted CD or DVD recordings fail with buffer underrun or data starvation errors, or (when recording CD formats) if you use underrated CD media (that is, CD media rated for speeds less than those of your CD recording device).

     You can use /SPEED to select the CD or DVD recording speed up to the I/O performance of the local OpenVMS system. Use of this qualifier is limited to the maximum recording speed ratings for the target drive and for the target recording media.

     Remember that /SPEED is more a go-slow than a go-fast mechanism. You need to choose to “go slow” when something goes wrong, such as when you use low-quality media or partially defective media.

   Differences exist between CD and DVD media in encoding a maximum speed:

   - CD media does not encode a maximum speed, although the media is manufactured with a maximum speed rating. Because there is no encoded limit, you can easily exceed the rated speed when recording.
   - DVD media encodes a maximum speed; the recording speed cannot exceed the rated limit for the media.

   Regardless of the media, other limits within the configuration can dictate a lower maximum recording speed; above this maximum speed, the recording operation fails.

   The /BELL qualifier specifies that a bell sound on completion of the operation.

   After the recording operation completes, the /VERIFY qualifier requests that OpenVMS read and compare the contents of the recorded media with the input data.

9. After successfully mastering your optical media, and you no longer need the particular LD logical disk for mastering, you can recover the disk storage occupied by the associated back-up storage file. Enter the following commands to disconnect and remove the LDA1: device from the system. You can then delete the back-up storage file:

   ```
   $ LD DISCONNECT LDA1:
   $ DELETE filespec.ISO;*
   ```

4.3 Examples

Example of Mastering Optical Media

The following sequence of commands shows how to create a 600 MB CD-R media recording on a CD-capable recording device, using an LD disk LDA600:. Its storage copy file is stored on the device DISK$SCRATCH:. The sequence assumes that a blank CD-R disk is loaded into the target DQA0: drive. A directory is created, and the user's LOGIN.COM file is copied into the directory. After recording is complete, the recorded media contains a single OpenVMS directory called [DATA].

```
$ @SYS$STARTUP:LD$STARTUP
$ LD CREATE DISK$SCRATCH:[000000]CD600.ISO/SIZE=1200000
$ LD CONNECT DISK$SCRATCH:[000000]CD600.ISO LDA600:
$ INITIALIZE/ERASE/SYSTEM LDA600: CDDATA
$ MOUNT/SYSTEM LDA600: CDDATA
$ CREATE/DIRECTORY LDA600:[DATA]
$ COPY SYS$LOGIN:LOGIN.COM LDA600:[DATA]
$ DISMOUNT LDA600:
$ COPY/RECORDABLE_MEDIA LDA600: DQA0:/VERIFY/BELL
$ LD DISCONNECT LDA600:
$ DELETE DISK$SCRATCH:[000000]CD600.ISO;
```

Example of Formatting and Recording Data onto a DVD

The following example shows how to format and then record the data stored on a LDA600: disk master onto a DVD+RW drive and its associated DVD+RW media. This media is located on device DQA0:.

```
$ COPY/RECORDABLE_MEDIA LDA600: DQA0:/FORMAT

HP OpenVMS CD-R/RW and DVD+R/RW Utility, V1.0-1 Copyright 1976,
2006 Hewlett-Packard Development Company, L.P.

Output device vendor: HL-DT-ST
Output device product name: DVD+RW GCA-4040N
Output device revision: 1.15
Commencing media format operation Formatting may require up to an hour
Output medium format: DVD+RW
Input data from: LDA600:
Writing data to: DQA0:
Input data size: 1200000 blocks

Starting operation at: 08:48:17

16 sectors written
```

```
30000 sectors written; estimated completion in 00:07:36; at 08:56:44 37000
sectors written; estimated completion in 00:07:37; at 08:56:57 46000
sectors written; estimated completion in 00:07:15; at 08:56:50 57000
sectors written; estimated completion in 00:06:43; at 08:56:34 71000
sectors written; estimated completion in 00:06:33; at 08:56:48 88000
sectors written; estimated completion in 00:05:56; at 08:56:39 110000
sectors written; estimated completion in 00:05:23; at 08:56:42 137000
sectors written; estimated completion in 00:04:37; at 08:56:41 171000
sectors written; estimated completion in 00:03:33; at 08:56:33 213000
sectors written; estimated completion in 00:02:23; at 08:56:32 266000
sectors written; estimated completion in 00:00:59; at 08:56:36 300000
sectors written; operation completed

Operation completed at: 08:56:32
Elapsed time for operation: 00:08:14
Synchronizing with output device cache
Processing completed
$
```

5
Programming Features

This chapter describes new features relating to application and system programming in this version of the HP OpenVMS operating system.

5.1 C Run-Time Library Enhancements

The following sections describe the C Run-Time Library (RTL) enhancements included in OpenVMS Version 8.3. These enhancements provide improved UNIX portability, standards compliance, and the flexibility of additional user-controlled feature selections. New C RTL functions are also included. For more details, refer to the *HP C Run-Time Library Reference Manual for OpenVMS Systems*.

5.1.1 Symbolic Link and POSIX-Compliant Pathname Support

OpenVMS Version 8.3 and higher provides Open Group-compliant symbolic-link support and POSIX-compliant pathname support. This support is intended to help partners and customers who port UNIX and LINUX applications to OpenVMS or who use a UNIX style development environment to reduce the application development costs and complexity previously associated with such porting efforts.

Although this support is present, it does not guarantee 100% compatibility of UNIX files on OpenVMS systems. There may be some cases where developers still need to make modifications to UNIX or LINUX applications when porting them to OpenVMS.

The following OpenVMS features are provided to support symbolic links and POSIX pathname processing:

- The following Open Group compliant symbolic-link functions are added to the C Run-Time Library:

  ```
  symlink
  readlink
  unlink
  realpath
  lchown
  lstat
  ```

- Existing C RTL functions such as `creat`, `open`, `delete`, and `remove`, now behave in accordance with Open Group specifications for symbolic links.
- RMS allows the C RTL to implement the above-mentioned functions. RMS routines such as SYS$OPEN, SYS$CREATE, SYS$PARSE, and SYS$SEARCH now support symbolic links.
- The contents of symbolic links on OpenVMS are interpreted as POSIX pathnames when encountered during pathwalks and searches. POSIX pathnames are now supported in OpenVMS and are usable through C RTL and RMS interfaces.

- A new feature logical DECC$POSIX_COMPLIANT_PATHNAMES is added to the C RTL to indicate that an application is operating in a POSIX-compliant mode.
- The DCL command CREATE/SYMLINK is used to create a symbolic link.
- The DCL command SET ROOT is provided to create the system POSIX root.
- Two GNV utilities, `mnt` and `umnt`, are provided to set mount points.
- DCL commands and utilities are modified to behave appropriately when acting on and encountering symbolic links.
- The TCP/IP Services for OpenVMS Network File System (NFS) client and server are enhanced to support symbolic links on ODS5 volumes.
- Relevant GNV utilities such as `ln` (which can create a symbolic link) and `ls` (which can display the contents of a symbolic link) are updated to provide access to and management of symbolic links.

For more information on symbolic links and POSIX pathname processing, see Chapter 12 of the *HP C Run-Time Library Reference Manual*.

5.1.2 Byte-Range Locking

The C RTL supports byte-range file locking using the F_GETLK, F_SETLK, and F_SETLKW commands of the `fcntl` function, as defined in the X/Open specification. The OpenVMS lock manager is used to implement this feature. Byte-range file locking is allowed across clusters. You can only use offsets that fit into 32-bit unsigned integers. For more information, see the `fcntl` function in the *HP C Run-Time Library Reference Manual*.

5.1.3 New C RTL Functions

In addition to the symbolic link functions listed in Section 5.1.1, the following new functions based on the X/Open specification have been added to the C RTL:

```
crypt
setkey
encrypt
fchmod
```

5.1.4 C RTL TCP/IP Header File Updates

The CRTL ships header files for users to call TCP/IP. These headers have had numerous problems, making some of them unusable for anything beyond trivial TCP/IP programming.

Previously, corrected headers have shipped with several releases of TCP/IP in their examples area. This enhancement to the C RTL now places those corrected headers in the C RTL header library (DECC$RTLDEF.TLB). For more information, see the C RTL section of the OpenVMS Version 8.3 Release Notes.

5.2 CDSA for OpenVMS and Secure Delivery

CDSA Version 2.2 for OpenVMS is based on the CDSA open source project. In addition, the CDSA implementation on OpenVMS is based on the Intel V2.0 Release 3 reference platform. New features in CDSA Version 2.2 for OpenVMS include Secure Delivery and support for HRS (Human Recognition Service Standard). These features are described in the following list.

- Support for HRS (Human Recognition Service Standard)

  CDSA Version 2.2 for OpenVMS contains support for HRS (Human Recognition Service). HRS is a CSSM (Common Security Services Manager) EMM (Elective Module Manager). It provides a generic authentication service that is suited for use with any form of human authentication (biometrics) for operation with CDSA.

  The HRS is designed for use by both application developers and biometric technology developers. It covers the basic functions of enrollment, verification, and identification, and includes a database interface to allow a biometric service provider (BSP) to manage the identification population for optimum performance.

- Secure Delivery

  Secure Delivery uses public key and digital signature technology to implement a system that provides OpenVMS users with the ability to authenticate and validate files from OpenVMS and third-party OpenVMS vendors.

  Secure Delivery allows you to create digital signatures for files, so that the file and associated manifest can be delivered over an unsecured channel such as the Internet or CD/DVD media. Once the files are in place on the target system, the manifest is used to authenticate the originator and validate the contents of the target file. If the target file or manifest has been tampered with in any way, the validation process fails. If the certificates used to sign the file have been revoked, the validation process fails.

  Secure Delivery has been integrated into PCSI, which automatically ensures that software installed on OpenVMS was not tampered with prior to installation. PCSI checks for the existence of a manifest for any kits that are being installed. If no manifest is found, PCSI issues a warning and asks if you want to proceed. If a manifest is found but does not match the kit, the installation is aborted. The PCSI database contains an indication as to whether a kit used Secure Delivery on installation.

  Most kits included on the OpenVMS Version 8.3 distribution media have been signed using Secure Delivery. On OpenVMS I64, layered product kits that have a manifest and are installed **during or after** the OpenVMS upgrade are validated. On OpenVMS Alpha, layered product kits that have a manifest and are installed **after** the OpenVMS upgrade are validated.

  Kits created before the Secure Delivery process was enabled in OpenVMS Version 8.3 can be installed on OpenVMS Version 8.3. These kits will be marked as unsigned, rather than as a validated kit in the PCSI history file. Products installed before Version 8.3 will have a blank validation status in the PCSI history.

For more information, refer to *HP Open Source Security for OpenVMS, Volume 1: Common Data Security Architecture*.

For additional information about CDSA, see the Common Data Security Architecture Web site at the following location:

```
http://sourceforge.net/projects/cdsa/
```

5.3 Deadlock Wait

OpenVMS V8.3 now supports the ability for a process to declare a sub-second deadlock wait time for the lock manager. This can be done with the $SET_PROCESS_PROPERTIES system service and the new item code PPROP$C_DEADLOCK_WAIT. The sub-second deadlock wait time overrides the system parameter DEADLOCK_WAIT time. In addition, the $GETJPI system service and F$GETJPI lexical function allow the retrieval of this time by using the item codes of JPI$_DEADLOCK_WAIT and DEADLOCK_WAIT. See the *HP OpenVMS System Services Reference Manual* and *HP OpenVMS DCL Dictionary* for more information on the usage.

The system parameter DEADLOCK_WAIT is in second units, so the smallest value you can set is 1 second. The sub-second deadlock wait time set via the system service $SET_PROCESS_PROPERTIES is valid only for the current image and is cleared during image rundown. The parameter passed is in 100nsec units and cannot exceed 1 sec. If the value is too small, then it is increased to a minimum value of 10msec. You can also call this system service to clear a previously set sub-second value by passing a parameter value of zero. Note the following example:

```
[...]
#define TEN_MSEC 100000

uint64 dead_wait;
uint64 prev_value;

 //
 // Set a 0.5 second deadlock wait time for the current process
 //
 dead_wait = 50 * TEN_MSEC;
 status = sys$set_process_properties ( 0, 0, 0, PPROP$C_DEADLOCK_WAIT, dead_wait, &prev_value );
[...]
```

5.4 Debugger New Features

The following sections describe new features of the OpenVMS Debugger on OpenVMS Version 8.3.

5.4.1 Improved C++ Support for Operator Names

Support has been improved for C++ operator names, now on Alpha systems as well as Integrity server systems. In particular, user-defined operator names are now supported. Prior to this change, you would have to enclose an operator name in %NAME.

For example:

```
DBG> SHOW SYMBOL /FULL operator ==
routine C::operator==
    type signature: bool operator==(C)
    code address: 198716, size: 40 bytes
    procedure descriptor address: 65752
DBG> SET BREAK operator==
```

5.4.2 Use of SET MODULE Command is Now Optional

Now available on Alpha systems as well as I64 systems, the OpenVMS Debugger can now set modules automatically, making use of an explicit SET MODULE command optional. There are two reasons for this feature:

- The debugger recognizes global symbol names and can determine from the symbol the module in which it is declared.
- The debugger automatically loads the module information when you specify the module name in a debugger command. For example, if you type:

  ```
  DBG> SET BREAK X\Y
  ```

  the debugger ensures that the module information for module X is loaded and then it locates the information for the routine named Y. Previously, the debugger complained that the information in module X was not loaded, whereas now, the debugger loads it automatically.

5.4.3 New Qualifier for SHOW STACK Command

Now on Alpha systems as well as Integrity server systems, the SHOW STACK command accepts a /START_LEVEL=*n* qualifier that directs SHOW STACK to begin displaying stack information at call frame level *n*.

For example, to see the stack information for only frame 3, type the following command:

```
DBG> SHOW STACK/START=3 1
```

To see the details for the 4th and 5th stack frames, type the following command:

```
DBG> SHOW STACK/START=4 2
```

5.4.4 Change to Default Data Type for Untyped Storage Locations

Now on Alpha systems as well as Integrity server systems, the default data type for untyped storage locations has been changed from longword (32 bits) to quadword (64 bits). Note that storage locations with data type information will continue to be displayed according to the related type.

5.4.5 Improved Overloaded Symbol Support in SHOW SYMBOL Command

Now on Alpha systems as well as Integrity server systems, the SHOW SYMBOL command has been enhanced to recognize overloaded symbol names. Previously, it just printed the various overloaded names. With the enhancement, it now prints each name and the associated information with that name. For example:

```
DBG> show symbol/full g
overloaded name C::g
    routine C::g(char)
        type signature: void g(char)
        address: 132224, size: 128 bytes
    routine C::g(long)
        type signature: void g(long)
        address: 132480, size: 96 bytes
```

5.4.6 GNAT Pro (Ada 95) Compiler Support Now Available on Integrity Server Systems (I64 Only)

The GNAT Pro (Ada 95) compiler is now supported on OpenVMS for Integrity servers systems. For information on this product, contact AdaCore directly.

Note that HP is not porting the HP Ada (Ada 83) compiler from OpenVMS Alpha to OpenVMS for Integrity servers.

5.4.7 Debugging Programs Loaded into P2 Space Now Supported

The OpenVMS Version 8.3 Debugger now allows you to debug programs loaded into P2 space.

5.4.8 SET WATCH Command Has Been Improved

Watchpoints on a location in memory (called static watchpoints) sometimes fail to notice writes by asynchronous system services and on occasion, might even cause failures of writes by asynchronous system services. For example, an asynchronous write by SYS$QIO to its IOSB output parameter may fail if that IOSB is being watched directly or even if it simply lives on the same page as an active static watchpoint.

The debugger now attempts to notice this condition. When it suspects a problem, the debugger warns the user about potential collisions between static watchpoints and asynchronous system services. For example:

```
DBG> g
%DEBUG-I-ASYNCSSWAT, possible asynchronous system service and static watchpoint collision
break at LARGE_UNION\main\%LINE 24192+60
DBG> sho call
 module name    routine name     line           rel PC           abs PC
*LARGE_UNION    main            24192       00000000000003A0 00000000000303A0
*LARGE_UNION    __main          24155       0000000000000110 0000000000030110
                                            FFFFFFFF80B90630 FFFFFFFF80B90630
DBG> ex/sour %line 24192
module LARGE_UNION
 24192:            sstatus = sys$getsyi (EFN$C_ENF, &sysid, 0, &syi_ile, &myiosb, 0, 0);
```

Type HELP MESSAGE ASYNCSSWAT in the debugger to learn more about the actions to take when this condition is detected.

5.4.9 Not a Thing (NaT) Support for Integer Registers

Previously, the debugger did not inform the user when an integer register’s NaT bit was set. The user had to examine the register’s bit in the %GRNAT0 register to learn this information. In the following example, integer registers R9 and R10 have their NaT bits set:

```
DBG> ex %r9:%r12
TEST\%R9:       0000000000000000
TEST\%R10:      0000000000000000
TEST\%R11:      0000000000000000
TEST\%SP:       000000007AC8FB70
DBG> ex/bin grnat0 <9,4,0>
TEST\%GRNAT0+1: 0110
DBG>
```

The debugger now displays the string "NaT" when the integer register’s NaT bit is set. For example:

```
DBG> ex %r9:%r12
TEST\%R9:       0000000000000000
TEST\%R10:      NaT
TEST\%R11:      NaT
TEST\%SP:       000000007AC8FB70
DBG> ex/bin grnat0 <9,4,0>
TEST\%GRNAT0+1: 0110
DBG>
```

Users can still see the actual physical value in a NaT register by specifying a type override. For example:

```
DBG> ex %r10
TEST\%R10:      NaT
DBG> ex/quad %r10
TEST\%R10:      0000000000000000
DBG>
```

5.4.10 Improved Debugger Usability: Automatic Module Loading Now Available

The debugger now automatically loads the symbol table for a module when the module names appear in a path name. Previously, a command like SET BREAK M\R, for example, would fail with an "unknown symbol R" error if the symbols for module M were not loaded. Now, the command succeeds; the debugger loads the symbols for module M first and is then able to locate the symbol R.

5.4.11 Improved Support for C++ Destructors

The debugger now recognizes C++ destructor names. Previously, the user was required to enclose the destructor name with the %NAME lexical. This is no longer required. The following example shows the new behavior:

```
DBG> examine C::~C
C::~C:             alloc       r35 = ar.pfs, 3F, 01, 00
DBG>
DBG> ex/source ~C
module CXXDOCEXAMPLE
    37:     ~C() {}
```

5.4.12 Support for C++ Template Names

The debugger now recognizes and supports C++ template names. For example:

```
DBG> e Map<string, int>::operator[]
Map<string, int>::operator[]:             alloc       r34 = ar.pfs, 1E, 05, 00
```

5.4.13 Improved Support for Ada Programs

The debugger now provides partial support for programs written in Ada. The debugger understands most Ada constructs, including packages, child units, procedures, variables, simple data types, and modular types. Support for more complicated data types, such as tagged types and pointers to unconstrained arrays, is expected in a future release.

5.5 Kerberos for OpenVMS

Kerberos Version 3.0 for OpenVMS is based on MIT Kerberos V5 Release 1.4.1. (The previous version of Kerberos, Version 2.1, was based on MIT Kerberos V5 Release 1.2.6.)

New features in Kerberos Version 3.0 are described in the following list.

- Upgrade to MIT Kerberos V5 Release 1.4.1

  Kerberos for OpenVMS Version 3.0 upgrades the code base to MIT Kerberos V5 Release 1.4.1. For a list of major and minor changes in each release of MIT Kerberos, see the Readme file for each release available from the following Web site:

  `http://web.mit.edu/kerberos/historical.html`

- Kerberos ACME Agent

  Kerberos for OpenVMS Version 3.0 includes the Kerberos ACME agent as part of an Advanced Developer's Kit (ADK). (This support is an addition to the existing Kerberos authentication provided by the Kerberos utilities that are a standard part of MIT Kerberos. It provides functionality similar to the pam_krb5 utility on UNIX systems using Kerberos.)

  In previous versions of OpenVMS, Kerberos for OpenVMS users were required to perform multiple login steps: once to log in to OpenVMS itself, and once to obtain Kerberos credentials. These steps worked with separate principal, or user, names and with separate passwords.

  To use the Kerberos ACME agent, install and setup the ACME Login Advanced Developer's Kit. See the SYS$HELP:ACME_DEV_README.TXT file for information about installation and set up. See the *HP Open Source Security for OpenVMS, Volume 3: Kerberos* for information about how to configure the Kerberos ACME agent.

  The user authentication will be processed against Kerberos´s KDC database instead of the OpenVMS UAF (User Authorization File). This new feature will give OpenVMS system managers additional flexibility: it will be possible to consolidate user databases so that multiple OpenVMS systems and clusters can be configured to automatically use a single KDC for user authentication.

- Support for AES Encryption

  Kerberos for OpenVMS now includes support for AES (Advanced Encryption Standard). AES is a symmetric key encryption technique which replaces the commonly used Data Encryption Standard (DES). In June 2003 the U.S. Government (NSA) announced that AES is secure enough to protect classified information up to the top secret level, which is the highest security level.

- Support for Kerberized SSH

  Kerberos for OpenVMS supports the Kerberized SSH functionality enabled in HP TCP/IP Services Version 5.6 for OpenVMS. Kerberized SSH allows you to use Kerberos credentials with your SSH (secure shell) connection. (SSH allows you to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another.)

- TCP Support in Client Libraries

  Kerberos for OpenVMS includes TCP support in client libraries. This is a Microsoft interoperability enhancement for tickets with a great deal of PAC data.

- Thread-safe KRB5 Libraries
- RPCSEC_GSS Authentication in the Kerberos RPC Library

Kerberos performs authentication as a trusted third-party authentication service by using conventional (shared secret key) cryptography. Kerberos provides a means of verifying the identities of principals, without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts on the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity.

For more detailed information, refer to *HP Open Source Security for OpenVMS, Volume 3: Kerberos*.

For information about downloading the latest version of Kerberos for OpenVMS, see the following Web site:

```
http://h71000.www7.hp.com/openvms/products/kerberos/
```

For additional information about Kerberos, see the MIT Kerberos Web site at the following location:

```
http://web.mit.edu/kerberos/www/
```

5.6 Linker Utility Enhancements

The object modules generated by C, C++, Ada, and possibly other compilers can have symbol names in the symbol table that have been altered; a process that is commonly referred to as “mangling”. These names are the symbol names visible to the linker, which the linker uses for symbol resolution.

The reason for mangling can be an overload feature in the programming language or simply the need to uniquely shorten names. When you link such modules and get an undefined-symbol message, the linker displays only the symbol name from the object module’s symbol table (that is, the mangled name). This processing makes it difficult to match the undefined, mangled symbol with the demangled, source code name. Therefore, to support upcoming compilers that produce display-name information (DNI), the linker can now process that information. In addition, the linker displays the source code name; that is, the linker can demangle the undefined symbol name. Further, if there is demangling information for universal symbols (that is, those to be exported from a shareable image), the linker can include that information in the generated shareable images.

The symbol resolution process remains unchanged. The linker still uses the mangled symbol names for symbol definitions and to resolve symbol references. The symbol vector option remains the same as well; it still requires the names found in the symbol tables—the mangled names.

You can use the new command-line qualifier, /DNI, to control the processing of the demangling information. Specify /DNI (the default) to allow the linker to attempt symbol-name demangling and move the necessary demangling information into the shareable image being created. Specify /NODNI when:

- You do not want the demangled names to be displayed.
- You do not want the demangling information to be moved into the shareable image.

You can request a new map section containing a translation table for the global symbol definitions. This table correlates the mangled symbol names with their demangled equivalents. By default, the linker does not generate this section in the map file. To request this section, specify the key word DEMANGLED_SYMBOLS to the /FULL qualifier. As with other keywords for the /FULL qualifier, specify the /MAP qualifier. Finally, specify the /DNI qualifier. The translation table in the map can be helpful to verify the symbol vector entries.

The following edited example shows how the demangled name is displayed in a linker message:

```
$ cre foo.cxx
extern double foo (int) ;
double bar (void) { return foo (123); }
Ctrl/Z
$ cxx foo
$ link foo
%ILINK-W-NUDFSYMS, 1 undefined symbol:
%ILINK-I-UDFSYM,        CXX$_Z3FOOI1RLFMIE
%ILINK-W-USEUNDEF, undefined symbol CXX$_Z3FOOI1RLFMIE referenced
        source code name: "double foo(int)"
        section: .text
        offset: %X0000000000000020  slot: 2
        module: FOO
        file: DISK$USER:[JOE]FOO.OBJ;1
$
```

The demangling information section is part of the object module or shareable image. The information needed to demangle the symbol name can be entirely contained within that section, or that information can include the name of a facilitator routine name and shareable image name to perform the demangling. This shareable image is usually located in SYS$LIBRARY and is provided by the language run-time environment already present on OpenVMS or is provided during the installation of the language compiler. If the facilitator routine is required, the linker attempts to activate the image and call the routine. If this fails at any point—for example, if the routine is not found in the shareable image or the image could not be located—then the linker displays an informational message indicating the problem. The link operation, as well as propagation of the demangling information into a shareable image, is independent of the success or failure of calling the facilitating routine.

5.7 Listing Demangled and Mangled Names with the Librarian (I64 Only)

The LIBRARY command now accepts the /DEMANGLED_SYMBOLS qualifier. When you use this qualifier, the Librarian lists symbols from an ELF object or ELF shareable image library that were altered by a language processor (also known as mangling) and prints their corresponding demangled name (that is, the name found in the source code). Mangled names are output as external symbols from an object module (or, as universal symbols from a shareable image). One reason for the language processor to mangle names is function overloading; a feature of the C++ language.

The Librarian stores the symbols that are emitted from an object module or shareable image. These symbols may include mangled symbols. The information needed to demangle these symbol names, along with a possible name of a facilitating shareable image, is contained in the object module or shareable image.

For an ELF object library, object modules are fully contained in the library. To retrieve the demangling information, the object module is mapped into memory and searched. It may be the case that language-specific demangling shareable images are also mapped into memory and activated. The symbols are read from the object library module and demangled with the information provided by the module. The library's symbol-name table is scanned to mark which of the object library module's symbols are in the library's symbol-name table. Subsequent modules are processed in the same way, and a listing is then generated.

For ELF shareable image libraries, the shareable image is not stored in the library. This is similar to OpenVMS Alpha and OpenVMS VAX shareable image libraries. Instead, only the exported symbols and other minimal information is stored in the library. As a result, the Librarian searches for the shareable image, external to the library, in order to obtain the demangling information.

The Librarian performs a three-step search; stopping when it successfully obtains a file:

1. First, the Librarian performs a logical name translation on the library module's name.
2. Failing that, the Librarian then searches for a module name with an extension of `.EXE` in the disk and directory in which the library resides.
3. If that fails, the Librarian finally looks in the device and directory indicated by the logical IA64$LIBRARY for a file with a filename of the library module and with a `.EXE` extension. If the Librarian finds a file specification, it maps it into memory and perform the same steps described above for ELF object libraries.

You can use the existing /OUTPUT= qualifier to direct the generated output from the demangling process to a file specification you choose. When you do not specify the /OUTPUT= qualifier, the Librarian by default places the output into a file in the current disk and directory, with the filename the same as the filename of the library and the extension `.LIS` as the file type.

You can also limit which library modules are selected for demangling by using the existing /ONLY= qualifier.

For example, to display the demangled symbols from an object library and have the output sent to the terminal screen, enter the following DCL command at the command line prompt:

```
$ LIBRARY /DEMANGLED_SYMBOLS /OUTPUT=SYS$OUTPUT OBJLIB.OLB
```

The following example captures the demangled symbol output to the file DUMP.LIS of the SHAREIMG.OLB shareable image library:

```
$ LIBRARY /DEMANGLED_SYMBOLS /OUTPUT=DUMP.LIS SHAREIMG.OLB
```

The following example captures the demangled symbol output to the default file specification SYS$DISK:[ ]SHAREIMG.LIS of the SHAREIMG.OLB shareable image library:

```
$ LIBRARY /DEMANGLED_SYMBOLS SHAREIMG.OLB
```

The following example captures the demangled symbol output to the default file specification SYS$DISK:[ ]OBJLIB.LIS of the object module MY_OBJ in the OBJLIB.OLB object library:

```
$ LIBRARY /DEMANGLED_SYMBOLS /ONLY=MY_OBJ OBJLIB.OLB
```

5.8 HP MACRO Compiler for OpenVMS Alpha Systems

The MACRO compiler has been upgraded to use the latest GEM backend for Alpha systems. In addition, several enhancements have been made:

- An /ARCHITECTURE qualifier has been provided. Possible values are GENERIC, HOST, EV4, EV5, EV56, EV6, EV67, and EV7. For architecture values of EV56 and later, the compiler now automatically generates Alpha byte/word instructions for corresponding VAX operations. In addition, the instruction scheduler will use the /ARCHITECTURE value as appropriate.
- The machine code listing has been improved including command line summary and symbolic names for most PAL calls.
- Three additional Alpha instruction builtins have been added:
  - EVAX_CTLZ <RQ,WQ> Generate a CTLZ instruction (count leading zeros)
  - EVAX_CTPOP <RQ,WQ> Generate a CTPOP instruction (count bits)
  - EVAX_CTTZ <RQ,WQ> Generate a CTTZ instruction (count trailing zeros) These new builtins are also accepted by the I64 compiler and equivalent Itanium instructions are generated.
- The name of the compiler image has been renamed to SYS$SYSTEM:MACRO.EXE. The prior SYS$SYSTEM:ALPHA_MACRO.EXE image has been left on the kit in the event that the new compiler produces incorrect results. The new compiler is used by OpenVMS engineering as the compiler used to build the system itself so there has already been extensive testing. The prior SYS$SYSTEM:ALPHA_MACRO.EXE compiler will be removed in a future release.

5.9 Record Management System (RMS) Enhancements

The following sections describe the RMS enhancements provided in OpenVMS Version 8.3.

5.9.1 RMS CONVERT/FDL and CREATE/FDL Enhancements

The CONVERT/FDL and CREATE/FDL routines have been enhanced to allow an FDL string to be passed as the FDL file specification. If the /FDL qualifier value is a quoted string and is not a quoted POSIX pathname, then it is passed to the FDL parser as an FDL string. Quoted values within the string must be designated with double quotes. See the *HP OpenVMS Utility Routines Manual* FDL section for details of using an FDL string.

Note the following examples:

```
$  CONVERT/FDL="TITLE ""This is an FDL string"";File;org SEQ;Record;size 80" -
_$( input_file:  output_file:)
```

```
$   CREATE/FDL="record;format fixed;size 100" file.dat
```

5.9.2 RMS Global Buffer Enhancements for Indexed Files

Prior to the OpenVMS Version 8.3 release, RMS global buffers were exclusively mapped to P0 (32-bit address) space. This restricted the overall per-process limit to less than 1 GB for the total number of global buffers specified for all files accessed by a process. This per-process limit forced RMS users to compromise on the number of global buffers specified for each file. It also constrained the per-file maximum size allowed for a global cache (currently, 32767 buffers).

To improve the overall scalability and performance of RMS global buffers, this release introduces the following global buffer enhancements for indexed files:

- Mapping RMS global buffers for indexed files to P2 (64-bit address) space in order to remove the overall per-process limit of less than 1 GB. No application changes are required to utilize global buffers for indexed files mapped to P2 space. However, global buffers must be enabled on an indexed file to take advantage of the enhancement.
- Increasing the per-file maximum size allowed for a global cache from a signed word (32767) to a signed longword (2.1 billion) for indexed files. To take advantage of the per-file maximum size increase, new options must be used with the SET FILE/GLOBAL_BUFFER qualifier.

Because RMS global buffers are local to each node, these enhancements were designed so they could be implemented on OpenVMS Alpha and OpenVMS for Integrity servers Version 8.3 nodes in a mixed cluster environment without requiring any changes to OpenVMS VAX or earlier version nodes. The Version 8.3 Alpha and Integrity server nodes will be able to increase the global cache size on an indexed file, while other nodes continue to operate on the file with the pre-Version 8.3 global cache size. This is particularly attractive, because it allows users to phase in Version 8.3 (or later) Alpha or Integrity server nodes that can support larger global buffer caches for indexed files into clusters with earlier versions of OpenVMS, including nodes running OpenVMS VAX.

5.9.3 New Form of Global Buffers Specification

There are now two forms of the SET FILE/GLOBAL_BUFFER command:

1. SET FILE/[NO]GLOBAL_BUFFER[=*n file name*]

   Where *n* sets the old value for the number of global buffers that can be shared (limited to a maximum of 32767). This makes the file compatible with prior versions of OpenVMS and stores in the original location in the file's header.
2. SET FILE/[NO]GLOBAL_BUFFER[=*keyword*[=*n*]] *file name*

   Where *keyword* can be:

   - COUNT=*n*—The value *n* sets the longword count of the number of global buffers.
   - PERCENT=*p*—The value *p* expresses the size of the global cache as a percent of the total number of blocks currently used in the file.
   - DEFAULT—Requests RMS at runtime to recalculate the global cache size based on an algorithm that makes use of two global buffer SYSGEN parameters, GB_CACHEALLMAX and GB_DEFPERCENT.

   The value *n* sets the new value stored in a different location in a file's header.

For example, the following commands do four distinctly different things:

1. `$ SET FILE/GLOBAL_BUFFER=20 NEWFILE.DAT ! Sets the compatibility (old) global buffer count (limited)`
2. `$ SET FILE/GLOBAL_BUFFER=COUNT=1000 NEWFILE.DAT ! Sets the new global buffer count (unlimited)`
3. `$ SET FILE/GLOBAL_BUFFER=PERCENT=50 INVENTORY.DAT ! Tells RMS to calculate the count as a percentage of the file`
4. `$ SET FILE/GLOBAL_BUFFER=DEFAULT INV.INX ! Tells RMS to calculate the count based on the total file size`

Global buffers specified with the old syntax (SET FILE/GLOBAL_BUFFER=*n*) stores settings in the file header in one location, while the new syntax (SET FILE/GLOBAL_BUFFER=*keyword*[=*n*]) are stored elsewhere in the file's header and are used to implement the new variation of global buffers, allowing more buffers and automatic adjustments for file growth.

Note You can specify only one version of the global buffer qualifier in a command string. Here's an example of using both the old and new global buffer specifications to set the old compatibility global buffer count value (for OpenVMS versions prior to Version 8.3) to 100 and the new value (Version 8.3 and later) to 10000:

```
$ SET FILE/GLOBAL_BUFFER=100 NEWFILE.DAT
$ SET FILE/GLOBAL_BUFFER=COUNT=10000 NEWFILE.DAT
```

In a clustered environment with mixed OpenVMS versions, the same file can be opened on different nodes with different global buffer counts. For nodes prior to Version 8.3, use the old compatibility setting, and for Version 8.3 nodes and later use the new values.

5.9.4 New Fields Added to XABFHC

Two new fields have been added to the read-only XABFHC (file header characteristics):

Field Offset	Bytes	Description
XAB$B_RECATTR_FLAGS	1	Flags for record attribute area
XAB$L_GBC32	4	Enhanced longword global buffer count

The field descriptions for these fields are the following:

- XAB$L_GBC32 — Enhanced longword global buffer count or percentage. If the XAB$V_GBC_PERCENT flag is set in the XAB$B_RECATTR_FLAGS field, the field contains a percentage.
- XAB$B_RECATTR_FLAGS — Flags for record attribute area in file header. The following flags are currently implemented:
  - XAB$V_GBC_PERCENT — This flag indicates the value in the global buffer count is expressed as a percent of the total number of blocks currently used in the file. This allows the size to dynamically grow over time because RMS recalculates the actual size to be mapped based on the current total number of blocks used in the file. The size is determined at run time each time the global cache is created by the first accessor on

a node. Users can also specify a percentage greater than 100 percent to allow for rapid growth for a file that, once opened, is closed infrequently.

— XAB$V_GBC_DEFAULT — This flag requests RMS at run time to recalculate the global cache size based on an algorithm that uses two global buffer (GB) SYSGEN parameters, GB_CACHEALLMAX and GB_DEFPERCENT. If the default option is enabled, and if the size (in blocks) of the file is less than or equal to the specified size for the GB_CACHEALLMAX parameter, RMS allocates sufficient global buffers to cache the whole file. If the size (in blocks) is greater than the specified size for the GB_CACHEALL MAX parameter, RMS allocates sufficient global buffers to cache the percentage of the file indicated by the GB_DEFPERCENT (global buffer default percent) parameter.

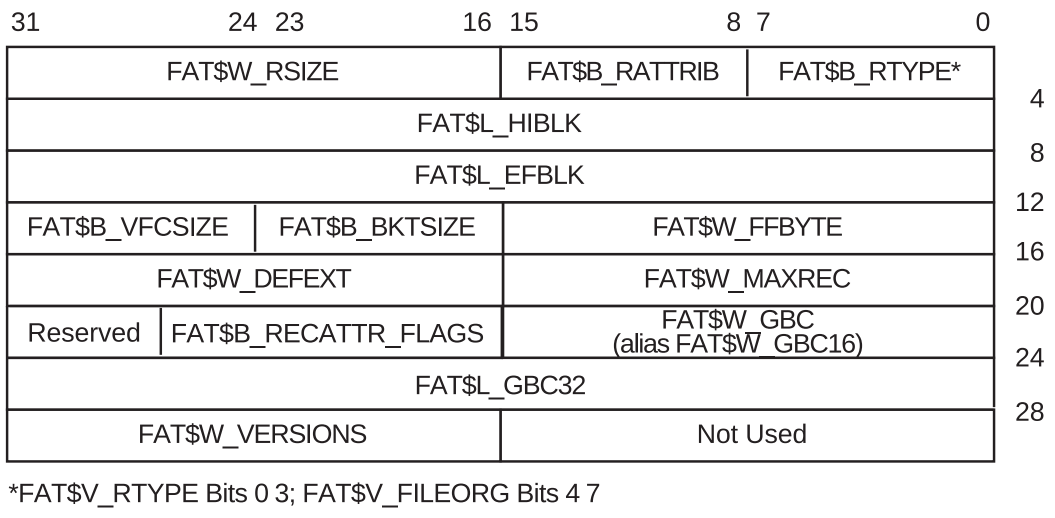
5.9.5 New RMS Field Values

The following field values have been added:

Field Value	Meaning
FAT$L_GBC32	Enhanced longword global buffer count
FAT$RECATTR_FLAGS	Record attributes flags. The following bit values are defined: FAT$M_GBC_PERCENT—Interpret value in FAT$L_GBC32 as a percent instead of count FAT$M_GBC_DEFAULT—RMS should set default for global buffer count and ignore any values in FAT$W_GBC or FAT$L_GBC32

Figure 5–1, from the *HP OpenVMS I/O User's Reference Manual*, has been updated to reflect the new fields.

Figure 5–1 ACP-QIO Record Attributes Area



ZK-0641-AI

5.9.6 New RMS Per-File Management Options for Sizing Global Buffer Cache

This release introduces two new per-file management options that allow simpler sizing of the global buffer cache for all RMS file organizations (sequential, relative, and indexed). These new options augment the existing global buffer count option:

- PERCENT — Rather than specifying a global buffer count as the size, users can express the size of the global cache as a percent of the current total number of blocks used in the file. This allows the size to dynamically grow over time, because RMS recalculates the actual size to be mapped based on the current total number of blocks used in the file. The size is determined at run time when the global cache is initially created by the first accessor of the file on a node. Users can also specify a percentage greater than 100 percent to allow for rapid growth for a file that, once opened, is closed infrequently.
- DEFAULT — The user can choose an option requesting that at run time each time the global cache is created by the first accessor on a node, RMS calculate a global cache size based on some file characteristics.

The RMS global buffer count (GBC) default is based on an algorithm that uses two new global buffer (GB) SYSGEN parameters: GB_CACHEALLMAX and GB_DEFPERCENT. If the default option is enabled, and if the size (in blocks) of the file is less than or equal to the size specified for the GB_CACHEALLMAX parameter, RMS allocates sufficient global buffers to cache the whole file.

If the size (in blocks) is greater than the specified size for the GB_CACHEALLMAX parameter, RMS allocates sufficient global buffers to cache the percentage of the file specified by the GB_DEFPERCENT (global buffer default percent) parameter. A percentage greater than 100 percent can be specified to provide space in the cache for the file to grow.

5.9.7 Size of Global Buffer Cache Connected to File (XAB$_GBC)

The XAB$_GBC item code can be used with an item list XAB on a $CONNECT or $DISPLAY with a SENSEMODE operation to sense the actual number of global buffers connected to a file when the global section for the file was created by the first accessor on the respective node.

The XAB$_GBC item code requires a 4-byte buffer to return the cache size that was connected. You cannot use a SETMODE with this item.

This option is not supported for DECnet operations; it is ignored.

5.9.8 Global Buffer Count (XAB$_GBC32)

The XAB$_GBC32 item code can be used with an item list XAB on a $CREATE with A SETMODE operation. It sets the longword global buffer count as a permanent attribute in the record-attribute area of the file header when a file is created. You can also use it with an item list XAB on a $CONNECT with a SETMODE to override at run time the permanent attribute in the file header. The override applies only if the process is the first connector of the cache on the respective node.

You cannot sense the global buffer count connected to a file using the XAB$_GBC32 item code. Use the XAB$_GBC item code to sense the actual global-buffer count.

The XAB$_GBC32 item code requires a 4-byte buffer to store the cache size that can be requested as either an actual count or a percentage of the total number of blocks in the file at run time. To specify the cache size as a percent, the XAB item list must also include the XAB$_GBCFLAGS item code; otherwise, the cache size value provided with the XAB$_GBC32 item code is interpreted as a count.

You can specify a maximum value of %x7FFFFFFF as the count for an indexed file; sequential and relative file organizations are restricted to a maximum of 32767. You can specify a percentage that is greater than 100 percent to allow for rapid growth for a file that, once opened, is closed infrequently.

Note that this option is not supported for DECnet operations; it is ignored.

5.9.9 Global Buffer Flags (XAB$_GBCFLAGS)

The XAB$_GBCFLAGS item code can be used with an item list XAB on a $CREATE with a SETMODE operation. It sets the global buffer flags value as a permanent attribute in the record attributes area of the file header when a file is created. You can also use it with an item list XAB on a $CONNECT with a SETMODE to override at run time the permanent attribute in the file header. The override applies only if the process is the first connector of the cache on the respective node.

You can sense the global buffer flags value using the XAB$_GBCFLAGS item code with an item list XAB on a $CONNECT or $DISPLAY with a SENSEMODE operation. The global buffer flags used in calculating the global buffer count when the global section for the file was created by the first connector is returned.

The two available flags are:

- XAB$M_GBC_PERCENT — Indicates the value in the global buffer count is expressed as a percent of the total number of blocks currently in the file. This allows the size to grow dynamically over time, because RMS recalculates the actual size to be mapped based on the current total number of blocks used in the file. The size is determined at run time each time the global cache is created by the first accessor on a node. You can also specify a percentage greater than 100 percent to allow for rapid growth for a file that, once opened, is closed infrequently.
- XAB$M_GBC_DEFAULT — Requests RMS at run time to recalculate the global cache size based on an algorithm that makes use of two global buffer (GB) SYSGEN parameters: GB_CACHEALLMAX and GB_DEFPERCENT. If the default option is enabled, and if the size (in blocks) of the file is less than or equal to the specified size for the GB_CACHEALLMAX parameter, RMS allocates sufficient global buffers to cache the whole file. If the size (in blocks) is greater than the specified size for the GB_CACHEALLMAX parameter, RMS allocates sufficient global buffers to cache the percentage of the file specified by the GB_DEFPERCENT (global buffer default percent) parameter.

The XAB$_GBCFLAGS item code requires a 4-byte buffer to store the flag value of either XAB$_GBC_PERCENT or XAB$_GBC_DEFAULT.

Note, this option is not supported for DECnet operations; it is ignored.

5.10 HP SSL for OpenVMS

Secure Sockets Layer (SSL) is the open standard security protocol for the secure transfer of sensitive information over the Internet. HP SSL Version 1.3 is based on OpenSSL 0.9.7e. (The previous version of HP SSL was based on OpenSSL 0.9.7d.)

Support for HP SSL is provided on OpenVMS I64, OpenVMS Alpha, and OpenVMS VAX.

HP SSL Version 1.3 is now included in the OpenVMS operating system as a SIP (system integrated product) instead of as a layered product. Version 1.3 also includes the bug fixes included in OpenSSL 0.9.7e. These features are described in the following list:

- SSL as SIP (System Integrated Product)

  SSL for OpenVMS is now installed automatically when you install or upgrade to OpenVMS Version 8.3. You no longer need to install the PCSI file separately.

- Bug Fixes in OpenSSL 0.9.7e

  The major changes between OpenSSL 0.9.7d and OpenSSL 0.9.7e are as follows:

  - Fixed race condition when CRLs are checked in a multithreaded environment.
  - Added Delta CRL to extension code.
  - Fixed s3_pkt.c so alerts are sent properly.
  - Reduced chances of duplicate issuer name and serial numbers (in violation of RFC3280) using the OpenSSL certificate creation utilities.

HP SSL addresses these three fundamental security concerns about communication over the Internet and other TCP/IP networks:

- SSL server authentication — Allows a user to confirm a server's identity. SSL-enabled client software can use standard techniques of public-key cryptography to check whether a server's certificate and public ID are valid and have been issued by a Certificate Authority (CA) listed in the client's list of trusted CAs. Server authentication is used, for example, when a PC user is sending a credit card number to make a purchase on the web and wants to check the receiving server's identity.
- SSL client authentication — Allows a server to confirm a user's identity. Using the same techniques as those used for server authentication, SSL-enabled server software can check whether a client's certificate and public ID are valid and have been issued by a Certificate Authority (CA) listed in the server's list of trusted CAs. Client authentication is used, for example, when a bank is sending confidential financial information to a customer and wants to check the recipient's identity.
- An encrypted SSL connection — Requires all information sent between a client and a server to be encrypted by the sending software and decrypted by the receiving software, thereby providing a high degree of confidentiality. Confidentiality is important for both parties to any private transaction. In addition, all data sent over an encrypted SSL connection is protected with a mechanism that automatically detects whether data has been altered in transit.

For more detailed information, refer to *HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS*.

For information about downloading the latest version of HP SSL for OpenVMS, see the following Web site:

`http://h71000.www7.hp.com/openvms/products/ssl/`

For additional information about OpenSSL, see the OpenSSL Web site at the following location:

`http://www.openssl.org/`

5.11 System Services New Information and New Item Codes

New item codes have been added for several system services. More information about some item codes has been added as well. These topics are discussed in the following sections.

5.11.1 $GETDVI: New Item Codes and Item Code Information

OpenVMS Version 8.3 contains the new item codes and item code information described in the following sections.

5.11.1.1 New $GETDVI Item Codes

New $GETDVI item codes are the following:

```
DVI$_DEVICE_MAX_IO_SIZE
DVI$_FC_HBA_FIRMWARE_REV
DVI$_LAN_ALL_MULTICAST_MODE
DVI$_LAN_AUTONEG_ENABLED
DVI$_LAN_DEFAULT_MAC_ADDRESS
DVI$_LAN_FULL_DUPLEX
DVI$_LAN_JUMBO_FRAMES_ENABLED
DVI$_LAN_LINK_STATE_VALID
DVI$_LAN_LINK_UP
DVI$_LAN_MAC_ADDRESS
DVI$_LAN_PROMISCUOUS_MODE
DVI$_LAN_PROTOCOL_NAME
DVI$_LAN_PROTOCOL_TYPE
DVI$_LAN_SPEED
DVI$_MAILBOX_BUFFER_QUOTA
DVI$_MAILBOX_INITIAL_QUOTA
DVI$_PREFERRED_CPU_BITMAP
DVI$_VOLUME_PENDING_WRITE_ERR
DVI$_VOLUME_RETAIN_MAX
DVI$_VOLUME_RETAIN_MIN
DVI$_VOLUME_SPOOLED_DEV_CNT
DVI$_VOLUME_WINDOW
```

5.11.1.2 $GETDVI Item Code Information

For item codes that return a string data type, failure to pass in a buffer that is large enough to hold the returned data results in silent data truncation. When $GETDVI completes, HP recommends that you check the returned length field of an item list descriptor for each item code that can return a string.

If the returned length is equal to the size of the buffer allocated to hold the returned data, the data might have been truncated. In that case, call $GETDVI iteratively with a larger buffer until the length of the returned data is less than the size of the buffer allocated.

Unless the description of an item code specifies otherwise, HP recommends that you use a buffer of 32 bytes to hold the returned string. $GETDVI pads the unused portion of the buffer with null characters.

5.11.2 $GETJPI New Item Code

The $GETJPI system service contains the new JPI$_DEADLOCK_WAIT item code.

5.11.3 $GETSYI New Item Codes

The $GETSYI system service contains the following new item codes:

```
SYS$_ACTIVE_CPU_BITMAP
SYI$_AVAIL_CPU_MASk
SYI$_BOOT_DEVICE
SYI$_IO_PRCPU_BITMAP
SYI$_POWERED_CPU_BITMAP
```

5.11.4 $GETDVI, $GETJPI, $GETLKI, $GETQUI, and $GETSYI Service Information

HP strongly recommends the use of the EFN$C_ENF “no event flag” value as the event flag if you are not using an event flag to externally synchronize with the completion of this system service call. The $EFNDEF macro defines EFN$C_ENF. For more information, see the *HP OpenVMS Programming Concepts Manual*.

5.11.5 $GETUAI New Item Codes

The $GETUAI system service contains the following new item codes:

```
UAI$V_DISPWDSYNCH
UAI$V_VMSAUTH
```

5.11.6 Additional Changes to System Services

Entries have been added or changed in $IO_FASTPATH and $PROCESS_AFFINITY related to the new CPU namespace project. Other additions and changes have been made to $SET_PROCESS_PROPERTIESW related to system service logging.

For more detailed information, see the *HP OpenVMS System Services Reference Manual*.

5.12 Traceback Facility

A callable interface that symbolizes program locations now exists on both OpenVMS Alpha and OpenVMS for Integrity servers. Previously, this interface was only available on OpenVMS for Integrity servers.

On Alpha systems, the new interface is called TBK$ALPHA_SYMBOLIZE, which is similar to the TBK$I64_SYMBOLIZE routine on Integrity server systems.

The Integrity server routine interface (TBK$I64_SYMBOLIZE) has changed from the previous release to match the Alpha routine interface (TBK$ALPHA_SYMBOLIZE) available in this release. This change is backwards-compatible; that is, the former interface is no longer documented but is supported for programs that currently use it.

Both interfaces support callers in USER, SUPER and EXEC mode. Previously on OpenVMS for Integrity servers, only USER mode callers were supported.

For complete information on the Traceback symbolize routines for Integrity servers and Alpha, see the *HP OpenVMS Utility Routines Manual*.

6
InfoServer Utility

This chapter provides a description of the InfoServer utility feature now supported on OpenVMS Alpha as well as OpenVMS for Integrity servers. This chapter includes a comparison between the InfoServer hardware and InfoServer application and a reference for the InfoServer utility commands.

6.1 InfoServer Utility Overview

The InfoServer application allows you to create a service for a virtual disk device on the local area network.

Virtual disk devices include the following:

- DVD drives
- Certain disk drives: SCSI and Fibre Channel
- CD drives
- Partitions (the equivalent of container files)

Comparison of InfoServer Hardware and the InfoServer Application

The new InfoServer application on OpenVMS differs from previous InfoServer hardware in a number of important ways. Some of the most notable are the following:

- The use of DCL-style command syntax
- The requirement that a device must be mounted before you can create a service for it
- Support for creating services for DVD drives
- No support for tape devices
- No support for CD-R (CD-recordable) drives.
- No automount support

6.1.1 InfoServer Usage Summary

You can use the InfoServer utility commands to do the following:

- Create and delete services for virtual disk devices on a local area network
- Save a list of active InfoServer services
- Modify the attributes of existing services
- Display information about servers and the nodes connected to services
- Display service-specific information about one or more services
- Start the LASTport/disk server and set various server and cache characteristics.

You can invoke the InfoServer in the following ways:

- **Use the RUN command**

  To invoke the InfoServer using the RUN command, enter the following at the DCL command prompt:

  ```
  $ RUN SYS$SYSTEM:ESS$INFOSERVER
  ```

  The system responds by displaying the InfoServer utility prompt. You can then enter an InfoServer command. For example:

  ```
  InfoServer> SHOW SERVER
  ```

  After the InfoServer executes the command, the system continues to display the InfoServer> prompt until you exit the utility.

- **Define the InfoServer as a foreign command**

  You can define the InfoServer as a foreign command by entering the following at the DCL prompt or in a startup or login command file:

  ```
  $ InfoServer :== $ESS$INFOSERVER
  ```

  After you execute the login command file, you can enter the INFOSERVER command at the DCL prompt to invoke the utility:

  ```
  $ INFOSERVER
  ```

  Note the following:

  - If you use InfoServer as a foreign command and also enter an InfoServer command, the utility terminates after it executes the command and returns you to the DCL command prompt. For example:

    ```
    $ InfoServer SHOW SERVER
    $
    ```

  - If you use InfoServer as a foreign command without specifying an InfoServer command, the utility displays the InfoServer> prompt, at which point you can enter commands. For example:

    ```
    $ InfoServer
    InfoServer>  SHOW SERVER
    ```

Note

All InfoServer commands require SYSPRV and OPER privileges.

To exit the InfoServer utility, enter the EXIT command at the InfoServer> prompt or press Ctrl/Z.

For information about the InfoServer utility, enter the HELP command at the InfoServer> prompt.

6.1.2 InfoServer Commands

The following sections describe and provide examples of InfoServer commands.

CREATE SERVICE

Creates a service for a specified device or partition.

Usage rules:

- All devices must be mounted systemwide to prevent them from being dismounted when a process logs out.
- A device that has read/write service must be mounted /FOREIGN so that it is not visible to OpenVMS.
- A device that has read-only service must be mounted with either the /NOWRITE qualifier or the /FOREIGN qualifier so that no one can change it locally.
- A partition can be served off a disk that is mounted for either read-only or read/write access to OpenVMS.
- Support for partitions is limited in this release.

Format

CREATE SERVICE   serviceName device-or-partitionName

Parameter

serviceName
The name by which the service is known to the local area network. The service name can consist of alphanumeric characters and dollar signs ($). It can be 255 characters or fewer in length.

device-or-partitionName
The device or partition name is the name of the OpenVMS disk device or partition as it is to be known to the local area network. The name of the device or partition that you enter must have been created previously.

Explanations of device and partition names follow.

- Device names

  Devices served to the local area network are OpenVMS disk devices; use OpenVMS device names when you specify an InfoServer device name. Note that the device name must either match exactly the name that the SHOW SERVICES command displays or must contain wildcards.

  In the InfoServer utility, wildcards, where supported, are the same as those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

  A disk specification must end with a colon.

- Partition names

  Partitions are container files that are served to the network. As such, they have OpenVMS file names with a default file type of .ESS$PARTITION. Partition names, including the device, directory, and file name, can be no more than 242 characters in length.

Support for partitions is limited in this version. HP strongly suggests that you use LD devices to support partitioned hard drives. See the DCL command LD HELP for more information.

Qualifiers

/CLASS=className

Specifies a subset of the complete LASTport Disk (LAD) name space.

The purpose of class names is to subdivide name spaces so that clients see only those names that are meaningful to them. The use of class names also allows two services to have the same name and not conflict with one another.

For example, you can use different class names for different on-disk structures that several client systems use. You might use SERVICEA/CLASS=ODS-2 for some client systems and SERVICEA/CLASS=ISO_9660 for other client systems. The service has the same name (SERVICEA), but the class names are different.

The class name you use depends on the client systems that will connect to the service being created. The default class name is ODS_2. For example, OpenVMS systems use the ODS_2 name space when attempting to mount an InfoServer device. Note that OpenVMS clients can solicit only those services that are in the ODS_2 service class.

Valid class names are the following:

V2.0	Names understood by PCSA MS-DOS Clients
Unformatted	Virtual disk has no format
MSDOS	MSDOS virtual disks
ODS_2	VMS virtual disks
UNIX	UNIX virtual disks
ISO_9660	ISO 9660 CD format
HIGH_SIERRA	MS-DOS CD format
APPLE	Macintosh HFS format
SUN	Sun format

/ENCODED_PASSWORD=hexstring

The SAVE command creates this qualifier. Because passwords are not stored in plain text, the hashed password value is written out as part of the SAVE operation so that the service can be recreated without revealing the password.

Note that if you edit the command procedure that the SAVE command creates and change the service name, the encoded password value is no longer valid. You need to set another password on the service using the /PASSWORD qualifier.

/PASSWORD=passwordString
/NOPASSWORD (default)

Specifies an optional service access control password. The client system must specify the password to access the service.

The password string can be up to 39 alphanumeric ASCII characters in length. If no password is specified, the client system is not required to provide a password to access the service.

The text password is hashed and stored in encrypted form in memory with the other service information.

/RATING=DYNAMIC
/RATING=STATIC=value

Clients use the service rating to select a service in the case of multiple matching services. The service with the highest service rating is selected.

The system adjusts the dynamic service rating based on load. You can also set a static rating between 0 and 65535. The system does not adjust static ratings.

One use of static ratings is to migrate clients from one copy of a service to another. If you set a static rating of 0 on services you want to migrate clients away from, no new clients will connect to a 0-rated service; instead, they will connect to higher-rated services. When all current clients have disconnected from a service, you can safely delete it.

/READAHEAD (default)
/NOREADAHEAD
When a disk read is required to fill a cache block, the /READAHEAD qualifier specifies that the read is to be from the first block requested to the end of the bucket boundary. Readahead can speed up sequential operations by preloading disk blocks that are needed into the cache.

If you specify both the /READAHEAD and the /READBEHIND qualifiers, any block requested within a cache bucket causes the entire bucket range of blocks to be read into the cache.

/READBEHIND
/NOREADBEHIND (default)
When a disk read is required to fill a cache block, the /READBEHIND qualifier specifies that the read is to include all blocks from the beginning of the cache bucket boundary up to and including the requested blocks.

If you specify both the /READAHEAD and the /READBEHIND qualifiers, any block requested within a cache bucket causes the entire bucket range of blocks to be read into the cache.

/READERS=number (default is READERS=1000)
/NOREADERS
Specifies the maximum number of simultaneous client connections allowed for read access. The default is 1000 readers. A value of 0 indicates write-only access.

If a client requests read-only or read/write access to a service, the system counts this as one reader.

/WRITERS
/NOWRITERS (default)
Specifies that the service is to allow access to a single writer.

Examples

1.
```
$ SHOW DEVICE MOVMAN$DQA0:/full

Disk MOVMAN$DQA0:, device type Compaq  CRD-8322B, is online, file-oriented
    device, shareable, served to cluster via MSCP Server, error logging is
    enabled.

 Error count                0    Operations completed
 Owner process             ""    Owner UIC                  [SYSTEM]
 Owner process ID    00000000    Dev Prot        S:RWPL,O:RWPL,G:R,W
 Reference count            0    Default buffer size             512
 Total blocks        16515072    Sectors per track                63
 Total cylinders        16384    Tracks per cylinder              16
```

```
$ MOUNT/SYSTEM dqa0 OVMSIPS11

Volume is write locked
OVMSIPS11 mounted on _MOVMAN$DQA0:

$ InfoServer
InfoServer> CREATE SERVICE VMS_SIPS_V11 _MOVMAN$DQA0:

%INFOSRVR-I-CRESERV, service VMS_SIPS_V11 [ODS-2] created for
_MOVMAN$DQA0:.
```

This example shows how to create a service for a CD device:

- The SHOW DEVICE . . . /FULL command displays a complete list of information about the _MOVMAN$DQA0 CD.
- The MOUNT/SYSTEM command mounts the OVMSIPS11 volume on the _MOVMAN$DQA0: CD.
- The InfoServer CREATE SERVICE command creates the VMS_SIPS_V11 service on the _MOVMAN$DQA0 CD.

2.
```
$ LD CREATE KIT1/SIZE-100000
$ DIRECTORY KIT1

Directory DKB0:[DISKS]

KIT1.DSK;1       100000/100008   29-APR-2005 14:14:43.49

Total of 1 file, 100000/100008 blocks.

$ LD CONNECT KIT1

%LD-I-UNIT, Allocated device is MOVMAN$LDA1:

 $ CREATE SERVICE TEST_KIT_1 MOVMAN$LDA1:

%INFOSRVR-I-CRESERV, service TEST_KIT_1 [ODS-2] created for
 _MOVMAN$LDA1:
```

This example shows how to create a service for a logical disk (LD) device:

- The LD CREATE KIT1 command creates a contiguous file, KIT1, that can be used as a logical disk.
- The DIRECTORY KIT1 command provides information about KIT1.
- The LD CONNECT KIT1 connects the logical disk file, KIT1, to the logical disk device MOVMAN$LDA1:.
- The INITIALIZE command formats the MOVMAN$LDA1: LD device.
- The MOUNT command makes the LD device available for processing.
- The CREATE SERVICE command creates the TEST_KIT_1 service on the _MOVMAN$LDA1 LD device.

DELETE SERVICE

Deletes one or more services.

Format

DELETE SERVICE serviceName [device-or-partitionName]

Parameters

serviceName
The name by which the service is known to the local area network. The service name can consists of alphanumeric characters and dollar signs ($). It can be up to and include 255 characters. Wildcards are permitted.

In the InfoServer utility, wildcards, where supported, are those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

device-or-partitionName
The device or partition name is the name of the OpenVMS disk device or partition as it is to be known to the local area network. The name of the device or partition that you enter must have been created previously.

Explanations of device and partition names follow.

- Device names

  Devices served to the local area network are OpenVMS disk devices; use OpenVMS device names when you specify an InfoServer device name. Note that the device name must either match exactly the name that the SHOW SERVICES command displays or must contain wildcards.

  In the InfoServer utility, wildcards, where supported, are those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

  A disk specification must end with a colon.

- Partition names

  Partitions are container files that are served to the network. As such, they have OpenVMS file names with a default file type of .ESS$PARTITION. Partition names, including the device, directory, and file name, can be no more than 242 characters in length.

  The partition name can be used to further identify the specific service selected.

  Support for partitions is limited in this version. HP strongly suggests that you use LD devices to support partitioned hard drives. See the DCL command LD HELP for more information.

Qualifiers

/CLASS=className
Specifies a subset of the complete LASTport Disk (LAD) name space.

The purpose of class names is to subdivide name spaces so that clients see only those names that are meaningful to them. The use of class names also allows two services to have the same name and not conflict with one another.

For example, you can use different class names for different on-disk structures that several client systems use. You might use SERVICEA/CLASS=ODS-2 for some client systems and SERVICEA/CLASS=ISO_9660 for other client systems. The service has the same name, SERVICEA, but the class names are different.

The class name you use depends upon the client systems that will connect to the service being created. The default class name is ODS_2. For example, OpenVMS systems use the ODS_2 name space when attempting to mount an InfoServer device. Note that OpenVMS clients can solicit only those services that are in the ODS_2 service class.

Valid class names are the following:

V2.0	Names understood by PCSA MS-DOS Clients
Unformatted	Virtual disk has no format
MSDOS	MSDOS virtual disks
ODS_2	VMS virtual disks
UNIX	UNIX virtual disks
ISO_9660	ISO 9660 CD format
HIGH_SIERRA	MS-DOS CD format
APPLE	Macintosh HFS format
SUN	Sun format

/CONFIRM (default)
/NOCONFIRM
Confirm the deletion of a service. If there are any connections, even though /NOCONFIRM has been entered, the system forces a confirmation.

Controls whether a request is issued before each delete operation to confirm that the operation should be performed on that service. The following responses are valid:

```
YES      NO       QUIT
TRUE     FALSE    Ctrl/Z
1        0        ALL
    Return (key)
```

Usage notes:

- You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE); however, these abbreviations must be unique.
- Affirmative answers are YES, TRUE, and 1. Negative answers include NO, FALSE, 0, and pressing Return.
- Entering QUIT or pressing Ctrl/Z indicates that you want to stop processing the command at that point.
- When you respond by entering ALL, the command continues to process, but no further prompts are displayed.

/DISCONNECT
/NODISCONNECT (default)
Overrides the default prompting for confirmation if you attempt to delete a service that has sessions connected to it. If a service has connected sessions and the /DISCONNECT qualifier is not supplied, you are prompted to confirm service deletion.

To delete services without being prompted at all, specify both the /NOCONFIRM and /DISCONNECT qualifiers.

Example

```
$ SHOW SERVICES

Service Name         [Service Class] Device or File
-------------------- --------------- --------------------------
HUDSON               [ODS-2]         _MOVERS$LDA1: [ 1 Connection]
BAFFIN               [ODS-2]         _MOVERS$LDA1:
FUNDY                [ODS-2]         _MOVERS$LDA1:
3 services found.

$ DELETE SERVICE HUDSON

Service HUDSON has 1 session connected!
Delete service HUDSON [ODS-2] for _MOVERS$LDA1:? [N]:
```

The first command displays three services, including one session connection. The second command deletes the HUDSON service. It displays messages indicating that HUDSON had one session connected and that this service has been deleted.

EXIT

Terminates the program. Alternatively, you can press Ctrl/Z to exit from the program.

Format

```
EXIT
```

HELP

Online InfoServer help.

ESS$INFOSERVER is the user interface for the LASTport/Disk server implemented as an application on OpenVMS. It is similar in behavior to the hardware InfoServer product although not identical to it.

Format

HELP [topic]

Parameters

topic
The topic for which help is requested.

Example

```
$ INFOSERVER HELP SHOW SESSIONS
```

This command displays help about the InfoServer command SHOW SESSIONS.

SAVE

Saves the current set of active services as a set of commands in a command procedure. You can then invoke the command procedure to reproduce the current services when you reboot the system.

Format

SAVE procedureName

Parameters

procedureName
Creates a command procedure that restores the current server state. The procedure name is the OpenVMS file name of the command procedure to be created. If you do not specify a file type, the type defaults to .COM.

The default procedure name is ESS$LAD_SERVICES.COM.

Example

```
$ SHOW SERVICES

   Service Name         [Service Class] Device or File
   -------------------- --------------- --------------
   BASELEVEL_A          [ODS-2]         _INFOS$LDA1:
   BASELEVEL_B          [ODS-2]         _INFOS$LDA2:
   BASELEVEL_C          [ODS-2]         _INFOS$LDA3:
   BASELEVEL_D          [ODS-2]         _INFOS$LDA4:

   FIELD_TEST_BASELEVEL [ODS-2]         _INFOS$LDA2:
   CURRENT_BASELEVEL    [ODS-2]         _INFOS$LDA3:
   EXPERIMENTAL_BASELEVEL
                        [ODS-2]         _INFOS$LDA4:
   %INFOSRVR-I-FOUND, 7 services found.

$ SAVE BASELEVELS
```

```
$! Created by the OpenVMS InfoServer SAVE command on 22-APR-2005
14:34:02.48
$ Set NoOn
$ Infoserver := $ESS$INFOSERVER
$!
$! The comment for each service includes the current device name.
$!
$!****************************************************************
$!  BASELEVEL_A [ODS_2] - _BILBO$LDA1: ❶
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_A.DSK;1 ❷
$ LD_UNIT_1 := LDA'LD_UNIT': ❸
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_1' BASELEVELA ❹
 $ INFOSERVER Create Service BASELEVEL_A 'LD_UNIT_1' - ❺
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  BASELEVEL_B [ODS_2] - _BILBO$LDA2:
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_B.DSK;1
$ LD_UNIT_2 := LDA'LD_UNIT':
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_2' BASELEVELB
$ INFOSERVER Create Service BASELEVEL_B 'LD_UNIT_2' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  BASELEVEL_C [ODS_2] - _BILBO$LDA3:
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_C.DSK;1
$ LD_UNIT_3 := LDA'LD_UNIT':
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_3' BASELEVELC
$ INFOSERVER Create Service BASELEVEL_C 'LD_UNIT_3' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  BASELEVEL_D [ODS_2] - _BILBO$LDA4:
$!****************************************************************
$ LD Connect/Symbol _BILBO$DKB0:[DISKS]BASELEVEL_D.DSK;1
$ LD_UNIT_4 := LDA'LD_UNIT':
$ If $STATUS Then Mount/System/NoWrite 'LD_UNIT_4' BASELEVELD
$ INFOSERVER Create Service BASELEVEL_D 'LD_UNIT_4' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic -
        /Encoded_Password=481C6B9081E742C2
           ! Invalid if service name changes ❻
$!****************************************************************
$!  FIELD_TEST_BASELEVEL [ODS_2] - _BILBO$LDA2:
$!****************************************************************
$ INFOSERVER Create Service FIELD_TEST_BASELEVEL 'LD_UNIT_2' - ❼
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$ INFOSERVER Create Service CURRENT_BASELEVEL 'LD_UNIT_3' -
        /Class=ODS_2/Readers=1000/NoWriters -
        /Readahead/NoReadbehind -
        /Rating=Dynamic
$!****************************************************************
$!  EXPERIMENTAL_BASELEVEL [ODS_2] - _BILBO$LDA4:
$!****************************************************************
$ INFOSERVER Create Service EXPERIMENTAL_BASELEVEL 'LD_UNIT_4' -
```

```
              /Class=ODS_2/Readers=1000/NoWriters -
              /Readahead/NoReadbehind -
              /Rating=Dynamic -
              /Encoded_Password=01F1D7374C0B81EC
                ! Invalid if service name changes ❽
      $ Exit
```

The SHOW SERVICES command in this example displays the services that are currently offered by the server. There is a set of software baselevels, each on its own logical disk and served to the LAN. The baselevels are labeled a through d, but, in addition, names help users so that they do not need to remember the corresponding letters.

Note that devices LDA2, LDA3, and LDA4 have two services assigned to each one.

The numbers in the example correspond to the numbers of the following explanations.

❶ The comment for each device contains the name of the device at the time the SAVE command was executed. LD devices are pseudodisk devices and might change unit numbers every time they are connected.

❷ This command connects an LD device to the container file and assigns the unit number to the DCL symbol LD_UNIT.

❸ A unique symbol is created for each device assigned to a container file.

❹ This command mounts the device specifying the label of the volume that the device had at the time of the SAVE command.

❺ The InfoServer service is re-created for the device.

❻ The experimental baselevel services are password protected. For security, the password is stored in the command procedure in prehashed format. Note that both services have the same password, but the hash is different.

❼ Because FIELD_TEST_BASELEVEL and BASELEVEL_B point to the same LD device, no attempt is made to create another device, and the correct unit (symbol LD_UNIT_2) is used to refer to the previously created unit.

❽ See explanation #6.

SET SERVICE

Modifies the attributes of an existing service.

Format

SET SERVICE serviceName [device-or-partitionName]

Parameters

serviceName
The name by which the service is known to the local area network. The service name can consist of alphanumeric characters or dollar signs ($). It can be up to 255 characters in length.

device-or-partitionName
The device or partition name is the name of the OpenVMS disk device or partition as it is to be known to the local area network. The name of the device or partition that you enter must have been created previously.

Explanations of device and partitions names follow.

- Device names

  Devices served to the local area network are OpenVMS disk devices; use OpenVMS device names when you specify an InfoServer device name. Note that the device name must either match exactly the name that the SHOW SERVICES command displays or must contain wildcards.

  In the InfoServer utility, wildcards, where supported, are those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

  A disk specification must end with a colon.

- Partition names

  Partitions are container files that are served to the network. As such, they have OpenVMS file names with a default file type of .ESS$PARTITION. Partition names, including the device, directory, and file name, can be no more than 242 characters in length.

  The partition name can be used to further identify the specific service selected.

  Support for partitions is limited in this version. HP strongly suggests that you use LD devices to support partitioned hard drives. See the DCL command LD HELP for more information.

Qualifiers

/CLASS=className
Specifies a subset of the complete LASTport Disk (LAD) name space.

The purpose of class names is to subdivide name spaces so that clients see only those names that are meaningful to them. The use of class names also allows two services to have the same name and not conflict with one another.

For example, you can use different class names for different on-disk structures that several client systems use. You might use SERVICEA/CLASS=ODS-2 for some client systems and SERVICEA/CLASS=ISO_9660 for other client systems. The service has the same name (SERVICEA), but the class names are different.

The class name you use depends upon the client systems that will connect to the service being created. The default class name is ODS_2. For example, OpenVMS systems use the ODS_2 name space when attempting to mount an InfoServer device. Note that OpenVMS clients can solicit only those services that are in the ODS_2 service class.

Valid class names are the following:

V2.0	Names understood by PCSA MS-DOS Clients
Unformatted	Virtual disk has no format
MSDOS	MSDOS virtual disks
ODS_2	VMS virtual disks
UNIX	UNIX virtual disks
ISO_9660	ISO 9660 CD format
HIGH_SIERRA	MS-DOS CD format
APPLE	Macintosh HFS format
SUN	Sun format

/PASSWORD=passwordString
/NOPASSWORD

Specifies an optional service access control password. The client system must specify the password to access the service.

The password string can be up to 39 alphanumeric ASCII characters in length. If no password is specified, the client is not required to provide a password to access the service.

The text password is hashed and stored in encrypted form in memory with the other service information.

/RATING=DYNAMIC
/RATING=STATIC=value

Clients use service rating to select a service in the case of multiple matching services. The service with the higher service rating is selected.

The system adjusts the dynamic service rating based on load.

A static rating between 0 and 65535 can also be set. Static ratings are not adjusted by the system.

/READAHEAD
/NOREADAHEAD

When a disk read is required to fill a cache lock, specifies that the read should be from the first block requested to the end of the bucket boundary. Readahead can speed up sequential operations by pre-loading disk blocks that are needed into the cache.

If both the /READAHEAD and the /READBEHIND qualifiers are specified, any block requested within a cache bucket causes the entire bucket range of blocks to be read into the cache.

/READBEHIND
/NOREADBEHIND
When a disk read is required to fill a cache block, specifies that the read should include all blocks from the beginning of the cache bucket boundary up to and including the requested block.

If both the /READAHEAD and the /READBEHIND qualifiers are specified, any block requested within a cache bucket causes the entire bucket range of blocks to be read into the cache.

/READERS=number
Specifies the maximum number of client connections allowed for read access.

Example

```
$ INFOSERVER SET SERVICE FUNDY/NOPASSWORD

   Service FUNDY [ODS-2] modified.

$ INFOSERVER SHOW SERVICES FUNDY/FULL

   FUNDY [ODS-2]                         Access: Read-only
     File or device: _MOVERS$LDA1: [750000 blocks]
     Flags: 00000000D2 {No Writers,Static Rating,Readbehind,Readahead}
     Rating:        Static,    42         Password:        Disabled
     Max Readers:            1000         Max Writers:            0
     Curr Readers:              0         Curr Writers:           0
     Reads:                     0         Writes:                 0
     Blocks Read:               0         Blocks Written:         0
```

The first command in this example modifies the FUNDY service so that the client does not need to enter a password to access the service. The second command displays the FUNDY service, showing that the use of a password has been disabled. (In the second example, notice that the use of a password is enabled for the FUNDY service.)

SHOW SERVER

Displays information about the server (that is, the system that provides services).

Format

SHOW SERVER

Example

```
$ INFOSERVER SHOW SERVER

   Node MOVERS [COMPAQ Professional Workstation XP1000] running OpenVMS XALD-
BL2

   LASTport/Disk Server Version 1.2

   Max Services:            64              Write Quota:          0
   Cache Buckets:         4096              Cache Bucket Size:   32 blocks
   Cache Size:        67108864 bytes
   Hits:                   478              Hit Percentage:      59%
   Misses:                 328

   Current Sessions:       0                Peak Sessions:          1

                        Read                      Write
   Requests:              40                          0
   Blocks:               319                          0
   Errors:                 0                          0
   Aborted:                0                          0
   Conflicts:              0                          0
```

This command displays information about the server that provides services to the client. The information displayed includes the following:

- The maximum number of services this server can offer simultaneously
- The current size of the cache
- Cache effectiveness statistics
- Current and maximum historical number of clients connected simultaneously
- I/O statistics

SHOW SERVICES

The SHOW SERVICES command displays service-specific information for one or all services offered by the server. This information includes the device number associated with the service and the number of connected sessions.

The SHOW SERVICES command supports wildcard expressions. In the InfoServer utility, wildcards, where supported, are those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

Format

SHOW SERVICES [serviceName] [options...]

Parameters

serviceName
The name by which the service is known to the local area network. The service name consists of alphanumeric characters or dollar signs ($). It can be up to 255 characters in length. If omitted, the service name defaults to ALL services.

In the InfoServer utility, wildcards, where supported, are the same as those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

Qualifiers

/BRIEF (default)
The BRIEF option provides an abbreviated one-line summary of information for each service selected. BRIEF is the default.

/FULL
The FULL option provides all the service-specific information for the services selected.

Examples

1. INFOSERVER> SHOW SERVICES

```
   Service Name         [Service Class] Device or File
   -------------------- --------------- -------------------------
   HUDSON                [ODS-2]         _MOVERS$LDA1:
   BAFFIN                [ODS-2]         _MOVERS$LDA1:
   FUNDY                 [ODS-2]         _MOVERS$LDA1:
   3 services found.
```

This command displays the one-line default BRIEF summary of all the services that are connected.

2.
```
INFOSERVER> SHOW SERVICES/FULL

HUDSON [ODS-2]                         Access: Read-only
  File or device: _MOVERS$LDA1: [750000 blocks]
  Flags: 0000000082 {No Writers,Readahead}
  Rating:       Dynamic, 65535        Password:        Disabled
  Max Readers:           1000        Max Writers:           0
  Curr Readers:             0        Curr Writers:          0
  Reads:                    0        Writes:                0
  Blocks Read:              0        Blocks Written:        0

BAFFIN [ODS-2]                        Access: Read-only
  File or device: _MOVERS$LDA1: [750000 blocks]
  Flags: 0000000082 {No Writers,Readahead}
  Rating:       Dynamic, 65535        Password:        Disabled
  Max Readers:           1000        Max Writers:           0
  Curr Readers:             0        Curr Writers:          0
  Reads:                    0        Writes:                0
  Blocks Read:              0        Blocks Written:        0

FUNDY [ODS-2]                         Access: Read-only
  File or device: _MOVERS$LDA1: [750000 blocks]
  Flags: 00000000D2 {No Writers,Static Rating,Readbehind,Readahead}
  Rating:        Static,    42        Password:        Enabled
  Max Readers:           1000        Max Writers:           0
  Curr Readers:             0        Curr Writers:          0
  Reads:                    0        Writes:                0
  Blocks Read:              0        Blocks Written:        0

3 services found.
```

This command displays all of the service-specific information for all the services that are connected. Notice that passwords are disabled on the HUDSON and BAFFIN services and enabled on the FUNDY service.

SHOW SESSIONS

Displays information about client nodes that are connected to services.

Format

SHOW SESSIONS [serviceName] [device-or-partitionName]]

Parameters

serviceName
The name by which the service is known to the local area network. The service name can consist of alphanumeric characters, dollar signs ($), and wildcards. It can be up to 255 characters in length. If omitted, the service name defaults to all services.

In the InfoServer utility, wildcards, where supported, are those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

device-or-partitionName
The device or partition name is the name of the OpenVMS disk device or partition as it is to be known to the local area network. The name of the device or partition that you enter must have been created previously.

Explanations of device and partition names follow.

- Device names

  Devices served to the local area network are OpenVMS disk devices; use OpenVMS device names when you specify an InfoServer device name. Note that the device name must either match exactly the name that the SHOW SERVICES command displays or must contain wildcards.

  In the InfoServer utility, wildcards, where supported, are the same as those used in OpenVMS. The percent (%) character matches exactly one character. The asterisk (*) character matches zero or more characters.

  A disk specification must end with a colon.

- Partition names

  Partitions are container files that are served to the network. As such, they have OpenVMS file names with a default file type of .ESS$PARTITION. Partition names, including the device, directory, and file name, can be no more than 242 characters in length.

  Support for partitions is limited in this version. HP strongly suggests that you use LD devices to support partitioned hard drives. See the DCL command LD HELP for more information.

Qualifiers

/ALL
Display all services that match the selection criteria even if no clients have connections. If this qualifier is omitted, only those services with clients connected are displayed.

Examples

1.
```
$ INFOSERVER SHOW SESSIONS
```
```
HUDSON              [ODS-2]         _MOVERS$LDA1: [ 1 Connection]
1 service found.
```

2.
```
$ INFOSERVER SHOW SESSIONS/ALL
```
```
HUDSON              [ODS-2]         _MOVERS$LDA1: [ 1 Connection]

BAFFIN              [ODS-2]         _MOVERS$LDA1:

FUNDY               [ODS-2]         _MOVERS$LDA1:
3 services found.
```

In the first example, this command displays only the session that has a client connection, HUDSON. In the second example, this command displays all sessions, even those with no client connections.

SPAWN

Spawns a process to execute a DCL command. If you do not enter a command, the command terminal is attached to the spawned process. If you do enter a command, that command is executed and, upon completion of the command, control returns to the parent process.

Format

SPAWN [DCL Command]

Example

```
InfoServer> SPAWN DIRECTORY

   .
   .
   .
(output)
   .
   .
   .

InfoServer>
```

This command spawns a process to execute a DCL command DIRECTORY. Following execution of the command, control returns to the InfoServer process.

START SERVER

This command starts the LASTport/Disk server and sets various server and cache characteristics.

Usually this command is executed by SYS$STARTUP:ESS$LAD_STARTUP.COM using data from SYS$STARTUP:ESS$LAD_STARTUP.DAT. HP strongly recommends that you make all modifications in the SYS$STARTUP:ESS$LAD_ STARTUP.DAT file.

You can use the START SERVER command interactively to use its qualifiers to change server settings as long as no services are currently defined.

Format

START SERVER

Qualifiers

/BUFFER_SIZE=n

The InfoServer block cache is structured as an array of fixed-size buffers (also called **buckets**.) The /BUFFER_SIZE qualifier determines the size of each bucket. (The /CACHE qualifier determines the number of buckets.)

The numeric value of this parameter is an integer between 3 and 8, inclusive, representing the bucket size in 512-byte blocks as follows:

```
   3 -   8 blocks (default)
   4 -  16 blocks
   5 -  32 blocks
   6 -  64 blocks
   7 - 128 blocks
   8 - 256 blocks
```

Bucket sizes that are larger than 32 blocks are not appropriate for most users. The OpenVMS client segments I/O requests that are larger than 31 blocks into 31-block chunks, and the default bucket readahead behavior might result in unnecessary I/O activity to the disk.

/CACHE = number-of-buckets (default = 512)

The InfoServer block cache is structured as an array of fixed-size buffers (also called **buckets**. The /CACHE qualifier determines the number of buckets in the cache. (The /BUFFER_SIZE qualifier determines the size of each bucket.)

Numbers larger than 16384 can adversely affect performance. Consider increasing the /BUFFER_SIZE qualifier to reach the desired cache size.

/MAXIMUM_SERVICES = maxservice (default = 256)

Sets the maximum service count for the server. This is the maximum number of services that can be defined at one time. Each service descriptor consumes nonpaged pool; however, unused service slots consume only 4 bytes each.

The maximum value is 1024.

/WRITE_QUOTA = n (default = 0)

Number of simultaneous synchronous writes permitted within the server. The default of zero means that all write operations are performed synchronously.

Example

```
$ InfoServer SHOW SERVER

   Node BILBO [HP rx2600  (900MHz/1.5MB)] running OpenVMS XAR8-D2Y
   LASTport/Disk Server Version 1.2

   Max Services:        64        Write Quota:          0
   Cache Buckets:     2048        Cache Bucket Size:   32 blocks
   Cache Size:    33554432 bytes
   Hits:                 0        Hit Percentage:       0%
   Misses:               0

   Current Sessions:     0        Peak Sessions:        0

                      Read             Write
   Requests:             0                 0
   Blocks:               0                 0
   Errors:               0                 0
   Aborted:              0                 0
   Conflicts:            0                 0

$ InfoServer START SERVER/MAXIMUM_SERVICES=128/CACHE=2048/BUFF=5/WRITE=0

   %INFOSRVR-I-STARTED, LASTport/Disk server started.

$ InfoServer SHOW SERVER

   Node BILBO [HP rx2600  (900MHz/1.5MB)] running OpenVMS XAR8-D2Y
   LASTport/Disk Server Version 1.2

   Max Services:    128        Write Quota:          0
   Cache Buckets:  2048        Cache Bucket Size:   32 blocks
   Cache Size: 33554432 bytes
   Hits:              0        Hit Percentage:       0%
   Misses:            0

   Current Sessions:  0        Peak Sessions:        0

                   Read             Write
   Requests:          0                 0
   Blocks:            0                 0
   Errors:            0                 0
   Aborted:           0                 0
   Conflicts:         0                 0
```

The first command in this example displays the current information about the server. The second command starts the server and increases the maximum number of services for the server. The third command displays the new information about the server, showing the increased number of maximum services.

7 Associated Products Features

This chapter describes significant new features of OpenVMS operating system associated products. For a listing and directory information about the OpenVMS associated products, refer to the *Read Before Installing* letter appropriate for your operating system.

7.1 Distributed NetBeans for OpenVMS

Distributed NetBeans for OpenVMS allows you to run the NetBeans IDE on your desktop system and develop applications on a remote OpenVMS Alpha or OpenVMS I64 system.

Distributed NetBeans Version 1.1 contains all of the functionality provided in NetBeans for OpenVMS plug-in modules: C/C++, COBOL, FORTRAN, and PASCAL language support, and MMS, BASH, DCL, CMS, and EDT keypad support. Distributed NetBeans supports access to your files and CMS libraries (including CMS groups) using a built-in FTP filesystem in addition to Samba for OpenVMS and Advanced Server.

For additional information about Distributed NetBeans for OpenVMS and to download the latest kits and documentation, see the following Web site:

```
http://www.hp.com/products/openvms/distributednetbeans/
```

7.2 Secure Web Browser for OpenVMS

Secure Web Browser Version 1.7-13 for OpenVMS is based on Mozilla M1.7.13 and includes important security bug fixes.

For additional information about the Secure Web Browser for OpenVMS and to download the latest kits and documentation, see the following Web site:

```
http://www.hp.com/products/openvms/securewebbrowser/
```

7.3 Secure Web Server for OpenVMS

Secure Web Server Version 2.1 for OpenVMS is based on Apache 2.0.52. New features include support for suEXEC and mod_dav, and the lifting of the STREAM_LF restriction present in Version 2.0.

Also available are new versions of PHP (CSWS_PHP Version 1.3), mod_perl (CSWS_PERL Version 2.1), Perl (PERL Version 5.8-6), and Tomcat (CSWS_JAVA Version 3.0) for use with the new Secure Web Server.

For additional information about the Secure Web Server for OpenVMS and to download the latest kits and documentation, see the following Web site:

```
http://www.hp.com/products/openvms/securewebserver/
```

7.4 HP TCP/IP Services for OpenVMS Version 5.6

HP TCP/IP Services for OpenVMS Version 5.6 supports OpenVMS Version 8.3. The following new features are provided in TCP/IP Version 5.6:

- BIND 9 resolver

  New version of the BIND resolver, including the ability to resolve DNS entries by way of the IPv6 transport. This represents a major upgrade from V5.5 and other recent releases, which provided resolver functionality based on BIND 8.

- DNS/BIND Version 9.3.1 server

  New version of the BIND server, which brings several incremental improvements related to security and stability.

- NFS client TCP support

  NFS client TCP support added, allows the NFS client as well as the server to run over TCP, in addition to the more-traditional UDP mode of operation. This can be useful when mounting file systems across a Wide Area Network or traversing a firewall.

- NFS Server support for Integrity servers

  Includes NFS server support for OpenVMS Integrity server platforms.

- NFS Symbolic Link Support

  Provides the ability for the NFS server to recognize symbolic links and create them as necessary.

- NTP security update (SSL)

  New NTP features offer cryptographic security, enhancing the protection against an attacker trying to compromise the accuracy of your system clock.

- SMTP multiple domains in a zone

  SMTP now recognizes more than one domain name for direct local delivery.

- SSH upgrade with Kerberos support

  TCP/IP Services Version 5.6 for OpenVMS introduces SSH support for Kerberos, the network authentication protocol from Massachusetts Institute of Technology. The SSH password authentication method has been enhanced to support Kerberos. Three new SSH authentication methods based on Kerberos are now supported:

  — gssapi-with-mic

  — kerberos-2@ssh.com

  — kerberos-tgt-2@ssh.com

  For more information about Kerberos, see to the *HP Open Source Security for OpenVMS, Volume 3: Kerberos*.

- TELNET upgrade with Kerberos support

  Support added for the TELNET server and client with the upgraded Kerberos version that ships with OpenVMS Version 8.3.

- TELNET Server Device Limit

  The TELNET server is no longer limited to 9999 sessions or TN devices.

- IPv6 support for LPD and TELNETSYM

  Both LPD and TELNETSYM printing software now allow you to print by way of the IPv6 transport.

- FTP performance enhancements for OpenVMS Plus Mode

  Streamlining was performed for the FTP service, specifically addressing the case where both server and client are OpenVMS systems.

- Improved interface configuration in TCPIP$CONFIG

  The menu-driven process of defining local interfaces and IP addresses has been significantly reworked to provide better support for failSAFE IP.

- Encryption for OpenVMS is now installed as part of the OpenVMS installation

  The menu-driven process of defining local interfaces and IP addresses has been significantly reworked to provide better support for failSAFE IP.

7.5 Web Services Integration Toolkit for OpenVMS

Web Services Integration Toolkit Version 1.1 for OpenVMS provides a set of individual tools to significantly help you develop a JavaBean to expose legacy application logic. These tools are designed to be valuable either individually or in combination.

The Web Services Integration Toolkit for OpenVMS provides tools that help you to:

- Create an XML IDL file

  Create an XML interface definition file (IDL) that describes the interface to be exposed.

- Generate components

  Generate a WSIT server interface wrapper and a WSIT JavaBean. Optionally, generate Java® or Java Server Page (JSP) clients.

- Use the generated code

  Call the generated WSIT JavaBean from the technology of your choice, including BEA Web Logic Server, Apache Axis, Java Message Service, Java Remote Method Invocation, Java Enterprise Edition (Java EE) or another JavaBean. Use the Java client with a command-line interface, or use the JSP client with a Web browser.

- Optionally, convert your existing BridgeWorks connections to an XML IDL file

  Optionally, convert your existing BridgeWorks connections to an XML IDL file that you can use with the Web Services Integration Toolkit.

For additional information about the Web Services Integration Toolkit for OpenVMS and to download the latest kits and documentation, see the following Web site:

```
http://www.hp.com/products/openvms/wsit/
```

Part II

OpenVMS Documentation

8 OpenVMS Documentation Overview

The OpenVMS Version 8.3 documentation contains fifteen revised manuals and four new release documents:

- Revised manuals
  - *HP OpenVMS Availability Manager User's Guide*
  - *HP C Run-Time Library Reference Manual for OpenVMS Systems*
  - *HP OpenVMS DCL Dictionary: A–M*
  - *HP OpenVMS DCL Dictionary: N–Z*
  - *HP OpenVMS Delta/XDelta Debugger Manual*
  - *HP OpenVMS Linker Utility Manual*
  - *HP OpenVMS Management Station Overview and Release Notes*
  - *HP Open Source Security for OpenVMS, Volume 1: Common Data Security Architecture*
  - *HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS*
  - *HP Open Source Security for OpenVMS, Volume 3: Kerberos*
  - *HP OpenVMS System Management Utilities Reference Manual: A–L*
  - *HP OpenVMS System Management Utilities Reference Manual: M–Z*
  - *HP OpenVMS System Services Reference Manual: A–GETUAI*
  - *HP OpenVMS System Services Reference Manual: GETUTC–Z*
  - *HP OpenVMS Utility Routines Manual*
- New release documents:
  - *HP OpenVMS Version 8.3 Upgrade and Installation Manual*
  - *HP OpenVMS Version 8.3 New Features and Documentation Overview*
  - *HP OpenVMS Version 8.3 Release Notes*
  - *Guide to HP OpenVMS Version 8.3 Media*

9 OpenVMS Printed and Online Documentation

OpenVMS documentation is provided, in the following ways:

- Printed documentation

  If you need paper documents, you can purchase most OpenVMS manuals in the form of printed documentation sets. Individual OpenVMS hardcopy documents cannot be purchased separately but are available in kits. One exception is the *Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers*, which you can order in hardcopy.

- Online documentation on CD

  All OpenVMS manuals are available in online formats on CD that also includes the documentation for many associated products. You automatically receive the documentation CD in your OpenVMS media kit.

- Online documentation on the OpenVMS documentation Web site

  You can preview or read any OpenVMS document, including archived manuals, on the OpenVMS Web site.

- Online help

  You can quickly display online help for OpenVMS commands, utilities, and system routines when you need task-related information.

The following sections describe each format in which OpenVMS documentation is provided and specifies the titles that are available in that format.

9.1 Printed Documentation

Some printed documentation comes with your OpenVMS Media Kit. All other printed manuals are orderable in kits. This section describes the OpenVMS printed documentation offerings, which are categorized as follows:

- Media kit
- Documentation sets:
  - Base
  - Full
  - Operating Environment Extensions
- System-integrated products
- Archived manuals

9.1.1 OpenVMS Media Kit Documentation

The OpenVMS Media Kit, for both OpenVMS Alpha and OpenVMS for Integrity servers systems, contains the documents you need to get started with the latest version of the OpenVMS operating system.

Table 9–1 lists the books included in the OpenVMS media kit. The books you receive are determined by whether you are a new or a service customer. New customers receive all books; service customers receive only new books and books that have been updated since the last release.

Note

The *HP OpenVMS License Management Utility Manual*, *Guide to HP OpenVMS Version 8.3 Media*, and *HP OpenVMS Version 8.3 Upgrade and Installation Manual* are provided only in the OpenVMS Media kit and, therefore, are not part of the OpenVMS Full Documentation set (described in Section 9.1.2).

Table 9–1 OpenVMS Media Kit Manuals

Manual	Order Number
HP OpenVMS License Management Utility Manual	AA-PVXUG-TK
Guide to HP OpenVMS Version 8.3 Media	BA322-90048
HP OpenVMS Version 8.3 New Features and Documentation Overview	BA322-90046
HP OpenVMS Version 8.3 Upgrade and Installation Manual	BA322-90045
HP OpenVMS Version 8.3 Release Notes	BA322-90047

9.1.2 OpenVMS Documentation Sets

OpenVMS documentation is available in the following documentation sets:

Documentation Set	Description	Alpha Order Number	Integrity Server Order Number
Full Set	Intended for users who need extensive explanatory information for all major OpenVMS resources. Contains all the OpenVMS documentation in one offering. Includes the Base Documentation set.	QA-001AA-GZ.8.3	BA554MN
Base Set	Subset of the Full Documentation set. Intended for general users and system managers of small standalone systems. Includes the most commonly used OpenVMS manuals.	QA-09SAA-GZ.8.3	BA555MN

There is one common documentation set for both OpenVMS Alpha and OpenVMS for Integrity servers systems. OpenVMS Alpha documentation set and the OpenVMS for Integrity servers documentation set contain the identical books with one exception. The OpenVMS Alpha documentation set contains the *COM, Registry, and Events for HP OpenVMS Developer's Guide*, which is an Alpha-only document. Table 9–2 lists the manuals in the OpenVMS Base and Full Documentation sets. For a description of each manual, see Section 10.2.

Table 9–2 OpenVMS Full Documentation Set (QA-001AA-GZ.8.3/BA554MN)

Manual	Order Number
OpenVMS Base Documentation Set	**QA-09SAA-GZ.8.3/BA555MN**
HP OpenVMS DCL Dictionary: A–M[1]	AA-PV5KL-TK
HP OpenVMS DCL Dictionary: N–Z[1]	AA-PV5LL-TK
HP OpenVMS Guide to System Security	AA-Q2HLH-TE
HP OpenVMS System Management Utilities Reference Manual: A–L[1]	AA-PV5PK-TK
HP OpenVMS System Management Utilities Reference Manual: M–Z[1]	AA-PV5QK-TK
HP OpenVMS System Manager's Manual, Volume 1: Essentials	AA-PV5MJ-TK
HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems	AA-PV5NJ-TK
OpenVMS User's Manual	AA-PV5JG-TK
HP OpenVMS Version 8.3 New Features and Documentation Overview[2]	BA322-90003
HP OpenVMS Version 8.3 Release Notes[2]	BA322-90004
Additional Books in the Full Documentation Set	**QA-001AA-GZ.8.3**
HP OpenVMS Availability Manager User's Guide[1]	AA-RNSJE-TE
COM, Registry, and Events for HP OpenVMS Developer's Guide[3]	AA-RSCWC-TE
HP C Run-Time Library Reference Manual for OpenVMS Systems[1]	AA-RSMUD-TE
Compaq C Run-Time Library Utilities Reference Manual	AA-R238C-TE
Compaq Portable Mathematics Library	AA-PV6VE-TE
DECamds User's Guide	AA-Q3JSE-TE
DEC Text Processing Utility Reference Manual	AA-PWCCD-TE
Extensible Versatile Editor Reference Manual	AA-PWCDD-TE
Guidelines for OpenVMS Cluster Configurations	AA-Q28LH-TK
Guide to Creating OpenVMS Modular Procedures	AA-PV6AD-TK
Guide to OpenVMS File Applications	AA-PV6PE-TK
Guide to the POSIX Threads Library	AA-QSBPD-TE
Guide to the DEC Text Processing Utility	AA-PWCBD-TE
HP Open Source Security for OpenVMS, Volume 1: Common Data Security Architecture[1]	AA-RSCUC-TE
HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS[1]	AA-RSCVD-TE
HP Open Source Security for OpenVMS, Volume 3: Kerberos[1]	AA-RUEBC-TE
HP OpenVMS Alpha Partitioning and Galaxy Guide	AA-REZQD-TE

[1]Revised for Version 8.3.
[2]New for Version 8.3.
[3]Alpha only - Provided only in QA-001AA-GZ.8.3

(continued on next page)

Table 9–2 (Cont.) OpenVMS Full Documentation Set (QA-001AA-GZ.8.3/BA554MN)

Manual	Order Number
Additional Books in the Full Documentation Set	**QA-001AA-GZ.8.3**
HP OpenVMS Guide to Upgrading Privileged-Code Applications	AA-QSBGE-TE
HP OpenVMS System Analysis Tools Manual	AA-REZTE-TE
HP OpenVMS Calling Standard	AA-QSBBE-TE
HP OpenVMS Cluster Systems	AA-PV5WF-TK
HP OpenVMS Command Definition, Librarian, and Message Utilities Manual	AA-QSBDE-TE
HP OpenVMS Debugger Manual	AA-QSBJE-TE
HP OpenVMS Delta/XDelta Debugger Manual[1]	AA-PWCADF-TE
HP OpenVMS I/O User's Reference Manual	AA-PV6SG-TK
HP OpenVMS Linker Utility Manual[1]	AA-PV6CDF-TK
HP OpenVMS MACRO Compiler Porting and User's Guide	AA-PV64DE-TE
HP OpenVMS Management Station Overview and Release Notes[1]	AA-QJGCH-TE
OpenVMS Performance Management	AA-R237C-TE
Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers	BA442-90001
HP OpenVMS Programming Concepts Manual, Volume I	AA-RNSHD-TK
HP OpenVMS Programming Concepts Manual, Volume II	AA-PV67H-TK
OpenVMS Record Management Services Reference Manual	AA-PV6RE-TK
OpenVMS Record Management Utilities Reference Manual	AA-PV6QD-TK
HP OpenVMS RTL General Purpose (OTS$) Manual	AA-PV6HE-TK
HP OpenVMS RTL Library (LIB$) Manual	AA-QSBHE-TE
OpenVMS RTL Screen Management (SMG$) Manual	AA-PV6LD-TK
OpenVMS RTL String Manipulation (STR$) Manual	AA-PV6MD-TK
OpenVMS System Messages: Companion Guide for Help Message Users	AA-PV5TD-TK
HP OpenVMS System Services Reference Manual: A–GETUAI[1]	AA-QSBMH-TE
HP OpenVMS System Services Reference Manual: GETUTC–Z[1]	AA-QSBNH-TE
HP OpenVMS Utility Routines Manual[1]	AA-PV6EG-TK
OpenVMS VAX RTL Mathematics (MTH$) Manual	AA-PVXJD-TE
OpenVMS VAX System Dump Analyzer Utility Manual	AA-PV6TD-TE
POLYCENTER Software Installation Utility Developer's Guide	AA-Q28MF-TK
VAX MACRO and Instruction Set Reference Manual	AA-PS6GD-TE
HP Volume Shadowing for OpenVMS	AA-PVXMK-TE

[1]Revised for Version 8.3.

9.1.3 Operating Environments Extensions Documentation Set (I64 Only)

The Operating Environments Extensions Documentation Set includes manuals that support the products that are included in the OEs. See Section 10.5 for a list of these documents.

9.1.4 Documentation for System Integrated Products

System Integrated Products (SIPs) are included in the OpenVMS software, but you must purchase separate licenses to enable them. Table 9–3 shows the documentation associated with System Integrated Products.

Table 9–3 System Integrated Products Documentation

System Integrated Product	Related Documentation
HP Galaxy Software Architecture on OpenVMS Alpha	The documentation is included in the OpenVMS Full Documentation Set.
OpenVMS Clusters	The OpenVMS Cluster documentation is included in the OpenVMS Full Documentation Set.
RMS Journaling for OpenVMS	RMS Journaling for OpenVMS manual is provided in HTML format on the OpenVMS Documentation Web site: `http://www.hp.com/go/openvms/doc`
Volume Shadowing for OpenVMS	The documentation is included in the OpenVMS Full Documentation Set.

9.1.5 Archived OpenVMS Documentation

OpenVMS continuously updates, revises, and enhances the OpenVMS operating system documentation. From time to time, manuals are archived. You can access the archived manuals online from the HP OpenVMS Version 8.3 Documentation CD or from the following Web site:

`http://www.hp.com/go/openvms/doc`

For a list of the archived OpenVMS manuals, see Section 10.6.

9.2 Authoring Tool for OpenVMS Documentation

OpenVMS Documentation team is continuing to introduce books that have been authored and published using a tool based on the Standard Generalized Markup Language (SGML). SGML is an industry standard and will provide many benefits to both the customer and OpenVMS documentation.

Readers will notice a difference in appearance between books produced from SGML and others in the documentation set. This is true for HTML, PDF, and printed formats and is a natural result of the new authoring environment.

The following Version 8.3 books have been produced with this new tool:

- *HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS*
- *HP Open Source Security for OpenVMS, Volume 3: Kerberos*
- *HP OpenVMS Version 8.3 Upgrade and Installation Manual*
- *Guide to HP OpenVMS Version 8.3 Media*
- *HP Open Source Security for OpenVMS, Volume 3: Kerberos*
- *HP OpenVMS I/O User's Reference Manual*
- *HP OpenVMS System Manager's Manual, Volume 1: Essentials*
- *HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems*

9.3 Online Documentation on CD

Online documentation for the OpenVMS operating system and many associated products is provided on one CD for both OpenVMS systems and Windows platforms. This CD is an ISO 9660 Level 2 CD that is readable on Windows® and OpenVMS systems.

9.3.1 Online Formats

The documentation CD contains documentation in the following formats:

Documentation	Available Formats
Current OpenVMS manuals	HTML, PDF
HP OpenVMS Version 8.3 Upgrade and Installation Manual	HTML, PDF
HP OpenVMS Version 8.3 Release Notes	HTML, PDF
HP OpenVMS Version 8.3 New Features and Documentation Overview	HTML, PDF
Layered product documents	HTML, PDF

Bookreader files are no longer available on the documentation CD.

For information about how to access documents on the documentation CD , see the *HP OpenVMS Version 8.3 Upgrade and Installation Manual*.

9.4 Online Documentation on the OpenVMS Web Site

You can access OpenVMS manuals in various online formats from the following OpenVMS Web site:

`http://www.hp.com/go/openvms/doc`

This site contains links to current versions of manuals in the OpenVMS Full Documentation Set as well as to manuals for selected layered products.

9.5 Online Help

The OpenVMS operating system provides online help for the commands, utilities, and system routines documented in the Full Documentation set.

You can use the Help Message facility to quickly access online descriptions of system messages. In addition, you can add your own source files, such as messages documentation that you have written to the Help Message database.

The *OpenVMS System Messages: Companion Guide for Help Message Users* manual explains how to use the Help Message facility. You can also access DCL Help for Help Message by entering:

```
$ HELP HELP/MESSAGE
```

Reference information for OpenVMS utility routines is also included in online help.

10
Descriptions of OpenVMS Manuals

This chapter provides summary descriptions for the following OpenVMS documentation:

- Manuals in the OpenVMS Media Kit (Section 10.1)
- Manuals in the OpenVMS Base and Full Documentation sets (Section 10.2 and Section 10.3)
- RMS Journaling manual (Section 10.4)
- Manuals in the OpenVMS for Integrity servers OE Extensions Kit
- Archived manuals (Section 10.6)

10.1 Manuals in the OpenVMS Media Kit

Guide to HP OpenVMS Version 8.3 Media

Provides information about the OpenVMS Version 8.3 operating system and documentation CD. Lists the contents of the OpenVMS Alpha and the OpenVMS for Integrity servers Version 8.3 media kits, includes pointers to installation information, and gives instructions about how to access manuals on the documentation CD.

HP OpenVMS License Management Utility Manual

Describes the License Management Facility (LMF), the OpenVMS license management tool. LMF includes the License Management Utility (LICENSE) and VMSLICENSE.COM, the command procedure you use to register, manage, and track software licenses.

HP OpenVMS Version 8.3 Upgrade and Installation Manual

Provides step-by-step instructions for installing the OpenVMS Alpha and OpenVMS for Integrity servers operating systems on their respective platforms. Includes information about booting, shutdown, backup, and licensing procedures.

HP OpenVMS Version 8.3 New Features and Documentation Overview

Describes new and improved components for the Integrity server and Alpha operating systems for the Version 8.3 release. Includes information about OpenVMS documentation changes for Version 8.3 as well as the printed and online OpenVMS documentation offerings.

HP OpenVMS Version 8.3 Release Notes

Describes changes to the software; installation, upgrade, and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

10.2 Manuals in the OpenVMS Base Documentation Set

HP OpenVMS DCL Dictionary

Describes the DIGITAL Command Language (DCL) and provides an alphabetical listing of detailed reference information and examples for all DCL commands and lexical functions. This manual is in two volumes.

HP OpenVMS Guide to System Security

Describes the security features available in the OpenVMS Alpha and VAX operating systems. Explains the purpose and proper application of each feature in the context of specific security needs.

HP OpenVMS System Management Utilities Reference Manual

Presents reference information about the utilities you can use to perform system management tasks on your system as well as the tools to control and monitor system access and resources. Includes a description of the AUTOGEN command procedure. This manual is in two volumes.

HP OpenVMS System Manager's Manual, Volume 1: Essentials

Provides instructions for setting up and maintaining routine operations such as starting up the system, installing software, and setting up print and batch queues. Also explains routine disk and magnetic tape operations.

HP OpenVMS System Manager's Manual, Volume 2: Tuning, Monitoring, and Complex Systems

Describes how to configure and control the network, how to monitor the system, and how to manage system parameters. Also includes information about OpenVMS Cluster systems, network environments, and DECdtm functionality.

OpenVMS User's Manual

Provides an overview of the operating system and presents basic concepts, task information, and reference information that allow you to perform daily computing tasks. Describes how to work with files and directories. Also includes these additional topics:

- Sending messages with the Mail utility and the Phone utility
- Using the Sort/Merge utility
- Using logical names and symbols
- Writing command procedures
- Editing files with the EVE and EDT text editors

HP OpenVMS Version 8.3 New Features and Documentation Overview

Describes new and improved components for the OpenVMS Alpha and OpenVMS for Integrity servers operating systems for the Version 8.3 release. Includes information about OpenVMS documentation changes for Version 8.3 as well as the printed and online OpenVMS documentation offerings.

HP OpenVMS Version 8.3 Release Notes

Describes changes to the software; installation, upgrade, and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

10.3 Additional Manuals in the OpenVMS Full Documentation Set

HP OpenVMS Availability Manager User's Guide

Describes how to use the HO Availability Manager system management tool, from either an OpenVMS Alpha or a Windows node, to monitor one or more OpenVMS nodes on an extended local area network (LAN) or to target a specific node or process for detailed analysis.

COM, Registry, and Events for HP OpenVMS Developer's Guide

For programmers developing applications that move easily between the OpenVMS and Windows NT environments. Read this manual if you are encapsulating existing OpenVMS applications or data, or creating new COM applications for OpenVMS systems. It also provides information for those who want to use the OpenVMS Registry to store information about their OpenVMS systems alone, or who want to use the OpenVMS Registry as a shared repository for both OpenVMS and Windows NT registry information. This manual was formerly available online as the *OpenVMS Connectivity Developer Guide*.

HP C Run-Time Library Reference Manual for OpenVMS Systems

Provides reference information on the functions and macros found in the HP C RTL that perform I/O operations, character and string manipulation, mathematical operations, error detection, subprocess creation, system access, and screen management. Includes portability concerns between operating systems, and describes the HP C for OpenVMS socket routines used for writing Internet application programs for the TCP/IP protocol.

Compaq C Run-Time Library Utilities Reference Manual

Provides detailed usage and reference information about the Run-Time Library utilities for managing localization and time zone data in international software applications.

Compaq Portable Mathematics Library

Documents the mathematics routines in the Compaq Portable Mathematics Library (DPML), supplied only with OpenVMS Alpha systems. VAX programmers should refer to the *OpenVMS VAX RTL Mathematics (MTH$) Manual*.

DECamds User's Guide

Provides information for installing and using the DECamds software. DECamds is a system management tool that lets you monitor, diagnose, and track events in OpenVMS system and OpenVMS Cluster environments.

DEC Text Processing Utility Reference Manual

Describes the DEC Text Processing Utility (DECTPU) and provides reference information about the EDT Keypad Emulator interfaces to DECTPU.

Extensible Versatile Editor Reference Manual

Contains command reference information about the EVE text editor. Also provides a cross-reference between EDT and EVE commands.

Guidelines for OpenVMS Cluster Configurations

This manual provides information to help you choose systems, interconnects, storage devices, and software. It can help you configure these components to achieve high availability, scalability, performance, and ease of system management. Detailed directions using SCSI and Fibre Channel in an OpenVMS Cluster system are also included in this manual.

Guide to Creating OpenVMS Modular Procedures

Describes how to perform a complex programming task by dividing it into modules and coding each module as a separate procedure.

Guide to OpenVMS File Applications

Contains guidelines for designing, creating, and maintaining efficient data files by using Record Management Services (RMS). This manual is intended for application programmers and designers responsible for programs that use RMS files, especially if performance is an important consideration.

Guide to the POSIX Threads Library

Describes the POSIX Threads Library (formerly named DECthreads) package, HP's multithreading run-time libraries. Use the routines in this package to create and control multiple threads of execution within the address space provided by a single process. Offering both usage tips and reference synopses, this document describes three interfaces: routines that conform to the IEEE POSIX 1003.1c standard (called *pthread*), routines that provide thread-related services in nonthreaded applications (called thread-independent services or *tis*), and a set of HP proprietary routines (called *cma*) that provide a stable, upwardly compatible interface.

Guide to the DEC Text Processing Utility

Provides an introduction to developing DECTPU programs.

HP Open Source Security for OpenVMS, Volume 1: Common Data Security Architecture

For application developers who want to use the Common Data Security Architecture (CDSA) to add security to their programs. Describes CDSA, gives information about installation and initialization, and provides example programs. Contains the CDSA application programming interface modules.

HP Open Source Security for OpenVMS, Volume 2: HP SSL for OpenVMS

For application developers who want to protect communication links to OpenVMS applications with HP Secure Sockets Layer (HP SSL) for OpenVMS. Contains installation instructions, release notes, and provides example programs. Includes programming information and a reference section for the OpenSSL application programming interface modules.

HP Open Source Security for OpenVMS, Volume 3: Kerberos

For application programmers who want to implement the Kerberos protocol that uses string cryptography, so that a client can proves identity to a server (and a server can provide its identity to a client) across an insecure network connection.

HP OpenVMS Alpha Partitioning and Galaxy Guide

Provides complete details about how to use all of the OpenVMS Galaxy features and capabilities available in OpenVMS Alpha Version 7.3–2. Includes procedures for creating, managing, and using OpenVMS Galaxy computing environments on AlphaServer 8400, 8200, and 4100 systems.

HP OpenVMS Guide to Upgrading Privileged-Code Applications

Explains the OpenVMS Alpha Version 7.0 changes that might impact Alpha privileged-code applications and device drivers as a result of the OpenVMS Alpha 64-bit virtual addressing and kernel threads support provided in OpenVMS Alpha Version 7.0.

Privileged-code applications from versions prior to OpenVMS Alpha Version 7.0 might require the source-code changes described in this guide.

HP OpenVMS System Analysis Tools Manual

Describes the following system analysis tools in detail, while also providing a summary of the dump off system disk (DOSD) capability and the DELTA/XDELTA debugger:

- System Dump Analyzer (SDA)
- System Code Debugger (SCD)
- System Dump Debugger (SDD)
- Watchpoint utility

Intended primarily for the system programmer who must investigate the causes of system failures and debug kernel mode code, such as a device driver.

HP OpenVMS Calling Standard

Documents the calling standard for the OpenVMS I64, Alpha, and VAX operating systems.

HP OpenVMS Cluster Systems

Describes procedures and guidelines for configuring and managing OpenVMS Cluster systems. Also describes how to provide high availability, building-block growth, and unified system management across clustered systems.

HP OpenVMS Command Definition, Librarian, and Message Utilities Manual

Contains descriptive and reference information about the following utilities:

- Command Definition utility
- Librarian utility
- Message utility

HP OpenVMS Debugger Manual

Explains the features of the OpenVMS Debugger for programmers.

HP OpenVMS Delta/XDelta Debugger Manual

Describes the Delta/XDelta utility used to debug programs that run in privileged processor mode or at an elevated interrupt priority level.

HP OpenVMS I/O User's Reference Manual

Contains the information that system programmers need to program I/O operations using the device drivers that are supplied with the operating system.

HP OpenVMS Linker Utility Manual

Describes how to use the Linker utility to create images that run on OpenVMS systems. Also explains how to control a link operation with link qualifiers and link options.

HP OpenVMS MACRO Compiler Porting and User's Guide

Describes how to port existing VAX MACRO assembly language code to an OpenVMS Alpha system by using the features of the MACRO-32 compiler. It also describes how to port existing OpenVMS Alpha code to OpenVMS I64 systems. Also documents how to use the compiler's 64-bit addressing support.

HP OpenVMS Management Station Overview and Release Notes

Provides an overview and release notes for OpenVMS Management Station and describes how to get started using the software. OpenVMS Management Station is a powerful, Microsoft Windows based management tool for system managers and others who perform user account and printer management tasks on OpenVMS systems.

OpenVMS Performance Management

Introduces and explains the techniques used to optimize performance on an OpenVMS system.

Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers

Provides a framework for application developers who are migrating from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers.

HP OpenVMS Programming Concepts Manual

Describes concepts such as process creation, kernel threads and the kernel threads process structure, interprocess communication, process control, data sharing, condition handling, and ASTs. This two-volume manual uses system services, utility routines, and run-time library (RTL) routines to illustrate mechanisms for utilizing OpenVMS features.

OpenVMS Record Management Services Reference Manual

Provides reference and usage information for all programmers who use RMS data files.

OpenVMS Record Management Utilities Reference Manual

Contains descriptive and reference information about the following RMS utilities:

- Analyze/RMS_File utility
- Convert and Convert/Reclaim utilities
- File Definition Language facility

HP OpenVMS RTL General Purpose (OTS$) Manual

Documents the general-purpose routines contained in the OTS$ facility of the OpenVMS Run-Time Library. Indicates which routines are specific to I64, Alpha or VAX, as well as how routines function differently on each system.

HP OpenVMS RTL Library (LIB$) Manual

Documents the general-purpose routines contained in the LIB$ facility of the OpenVMS Run-Time Library. Indicates which routines are specific to I64, Alpha or VAX, as well as how routines function differently on each system.

OpenVMS RTL Screen Management (SMG$) Manual

Documents the screen management routines contained in the SMG$ facility of the OpenVMS Run-Time Library. Indicates which routines are specific to Alpha or VAX, as well as how routines function differently on each system.

OpenVMS RTL String Manipulation (STR$) Manual

Documents the string manipulation routines contained in the STR$ facility of the OpenVMS Run-Time Library. Indicates which routines are specific to Alpha or VAX, as well as how routines function differently on each system.

OpenVMS System Messages: Companion Guide for Help Message Users

Describes features of the Help Message facility, a tool that you can use to display message descriptions. Describes the HELP/MESSAGE command and qualifiers and also includes detailed information about customizing the Help Message database. Also provides descriptions of messages that can occur when the system and Help Message are not fully operable.

HP OpenVMS System Services Reference Manual

Presents the set of routines that the operating system uses to control resources, allow process communication, control I/O, and perform other such operating system functions. This manual is in two volumes.

HP OpenVMS Utility Routines Manual

Describes the routines that allow a program to use the callable interface of selected OpenVMS utilities.

OpenVMS VAX RTL Mathematics (MTH$) Manual

Documents the mathematics routines contained in the MTH$ facility of the OpenVMS Run-Time Library, which is relevant only to programmers using OpenVMS VAX. (Alpha programmers should refer to *Compaq Portable Mathematics Library*.)

OpenVMS VAX System Dump Analyzer Utility Manual

Explains how to use the System Dump Analyzer utility to investigate system failures and examine a running OpenVMS VAX system. VAX programmers should refer to this manual; Alpha and I64 programmers should refer to the *OpenVMS Alpha System Dump Analyzer Utility Manual*.

POLYCENTER Software Installation Utility Developer's Guide

Describes the procedure and provides guidelines for developing software products that will be installed using the POLYCENTER Software Installation utility. Intended for developers who are designing installation procedures for software products layered on the OpenVMS operating system.

VAX MACRO and Instruction Set Reference Manual

Documents both the assembler directives of VAX MACRO and the VAX instruction set.

HP Volume Shadowing for OpenVMS

Describes how to provide high data availability with phase II volume shadowing.

10.4 RMS Journaling Manual

RMS Journaling for OpenVMS Manual

Describes the three types of RMS Journaling as well as other OpenVMS components that support RMS Journaling. This manual also describes the RMS Recovery utility (which is used to recover data saved using journaling), the transaction processing system services, and system management tasks required when using RMS Journaling.

10.5 Manuals in the OpenVMS for Integrity Servers OE Extensions Kit

The following list contains manuals relevant to the OpenVMS I64 Operating Environments.

- *HP DECwindows Motif for OpenVMS Installation Guide*
- *HP DECwindows Motif for OpenVMS New Features*
- *HP DECwindows Motif for OpenVMS Documentation Overview*
- *HP DECwindows Motif for OpenVMS Management Guide*
- *HP DECnet-Plus for OpenVMS Installation and Configuration*
- *HP DECnet-Plus for OpenVMS Introduction and User's Guide*
- *HP DECnet-Plus Network Management*
- *HP DECnet-Plus for OpenVMS DECdts Programming Reference*
- *HP DECnet-Plus for OpenVMS DECdts Management*
- *HP DECnet-Plus for OpenVMS DECdns Management*
- *HP DECnet-Plus for OpenVMS Network Management Quick Reference Guide*
- *HP DECnet-Plus for OpenVMS OSAK Programming*
- *HP DECnet-Plus for OpenVMS OSAK Programming Reference*
- *HP DECnet-Plus for OpenVMS OSAK SPI Programming Reference*
- *HP DECnet-Plus for OpenVMS Problem Solving Manual*
- *HP DECnet-Plus for OpenVMS Programming Manual*
- *HP DECnet-Plus for OpenVMS FTAM and Virtual Terminal User and Management*
- *HP DECnet-Plus for OpenVMS Problem Solving*
- *HP DECnet-Plus for OpenVMS Network Control Language Reference*
- *HP DECnet-Plus for OpenVMS Planning Guide*
- *HP TCP/IP Services for OpenVMS Installation and Configuration*
- *HP TCP/IP Services for OpenVMS Sockets API and System Services Programming*
- *HP TCP/IP Services for OpenVMS Concepts and Planning*
- *HP TCP/IP Services for OpenVMS SNMP Programming Reference*
- *HP TCP/IP Services for OpenVMS ONC RPC Programming*
- *HP TCP/IP Services for OpenVMS Tuning and Troubleshooting*
- *HP TCP/IP Services for OpenVMS Guide to SSH for OpenVMS*
- *HP TCP/IP Services for OpenVMS Management*
- *HP TCP/IP Services for OpenVMS Management Command Reference*
- *HP TCP/IP Services for OpenVMS Management Command Quick Reference Card*
- *HP TCP/IP Services for OpenVMS User's Guide*

- *HP TCP/IP Services for OpenVMS UNIX Command Equivalents Reference Card*
- *HP TCP/IP Services for OpenVMS Guide to IPv6*
- *HP DECprint Supervisor (DCPS) for OpenVMS User's Guide*
- *HP DECprint Supervisor (DCPS) for OpenVMS Software Installation*
- *HP DECprint Supervisor (DCPS) for OpenVMS Manager's Guide*
- *HP DCE for OpenVMS Product Guide*
- *HP DCE for OpenVMS Reference Guide*
- *HP DCE for OpenVMS Installation and Configuration Guide*

10.6 Archived Manuals

Table 10–1 lists the OpenVMS manuals that have been archived. Note that most information from the archived manuals has been incorporated in other documents or online help.

Table 10–1 Archived OpenVMS Manuals

Manual	Order Number
A Comparison of System Management on OpenVMS AXP and OpenVMS VAX	AA-PV71B-TE
Building Dependable Systems: The OpenVMS Approach	AA-PV5YB-TE
Creating an OpenVMS Alpha Device Driver from an OpenVMS VAX Device Driver	AA-R0Y8A-TE
Creating an OpenVMS AXP Step 2 Device Driver from a Step 1 Device Driver	AA-Q28TA-TE
Creating an OpenVMS AXP Step 2 Device Driver from an OpenVMS VAX Device Driver	AA-Q28UA-TE
Guide to OpenVMS AXP Performance Management	AA-Q28WA-TE
Guide to OpenVMS Performance Management	AA-PV5XA-TE
Migrating an Application from OpenVMS VAX to OpenVMS Alpha	AA-KSBKB-TE
Migrating an Environment from OpenVMS VAX to OpenVMS Alpha	AA-QSBLA-TE
Migrating to an OpenVMS AXP System: Planning for Migration	AA-PV62A-TE
Migrating to an OpenVMS AXP System: Recompiling and Relinking Applications	AA-PV63A-TE
OpenVMS Alpha Guide to 64-Bit Addressing and VLM Features	AA-QSBCC-TE
OpenVMS Alpha System Dump Analyzer Utility Manual	AA-PV6UC-TE
OpenVMS Alpha Version 7.3–1 New Features and Documentation Overview	AA-RSHYA-TE
OpenVMS Alpha Version 7.3–1 Release Notes	AA-RSD0A-TE
OpenVMS AXP Device Support: Developer's Guide	AA-Q28SA-TE
OpenVMS AXP Device Support: Reference	AA-Q28PA-TE
OpenVMS Bad Block Locator Utility Manual	AA-PS69A-TE

(continued on next page)

Table 10–1 (Cont.) Archived OpenVMS Manuals

Manual	Order Number
OpenVMS Compatibility Between VAX and Alpha	AA-PYQ4C-TE
OpenVMS Developer's Guide to VMSINSTAL	AA-PWBXA-TE
OpenVMS DIGITAL Standard Runoff Reference Manual	AA-PS6HA-TE
OpenVMS EDT Reference Manual	AA-PS6KA-TE
OpenVMS Exchange Utility Manual	AA-PS6AA-TE
OpenVMS Glossary	AA-PV5UA-TK
OpenVMS Guide to Extended File Specifications	AA-REZRB-TE
OpenVMS Master Index	AA-QSBSD-TE
OpenVMS National Character Set Utility Manual	AA-PS6FA-TE
OpenVMS Obsolete Features Manual	AA-PS6JA-TE
OpenVMS Programming Environment Manual	AA-PV66B-TK
OpenVMS Programming Interfaces: Calling a System Routine	AA-PV68B-TK
OpenVMS RTL DECtalk (DTK$) Manual	AA-PS6CA-TE
OpenVMS RTL Parallel Processing (PPL$) Manual	AA-PV6JA-TK
OpenVMS Software Overview	AA-PVXHB-TE
OpenVMS SUMSLP Utility Manual	AA-PS6EA-TE
OpenVMS System Messages and Recovery Procedures Reference Manual: A–L	AA-PVXKA-TE
OpenVMS System Messages and Recovery Procedures Reference Manual: M–Z	AA-PVXLA-TE
OpenVMS Terminal Fallback Utility Manual	AA-PS6BA-TE
OpenVMS VAX Card Reader, Line Printer, and LPA11–K I/O User's Reference Manual	AA-PVXGA-TE
OpenVMS VAX Device Support Manual	AA-PWC8A-TE
OpenVMS VAX Device Support Reference Manual	AA-PWC9A-TE
OpenVMS VAX Patch Utility Manual	AA-PS6DA-TE
OpenVMS Wide Area Network I/O User's Reference Manual	AA-PWC7A-TE
PDP-11 TECO User's Guide	AA-K420B-TC
POLYCENTER Software Installation Utility User's Guide	AA-Q28NA-TK
TCP/IP Networking on OpenVMS Systems	AA-QJGDB-TE
Standard TECO Text Editor and Corrector for the VAX, PDP-11, PDP-10, and PDP-8	Available only on CD

Table 10–2 lists the networking manuals and installation supplements that have been archived.

Table 10–2 Archived Networking Manuals and Installation Supplements

Manual	Order Number
DECnet for OpenVMS Guide to Networking	AA-PV5ZA-TK

(continued on next page)

Table 10–2 (Cont.) Archived Networking Manuals and Installation Supplements

Manual	Order Number
DECnet for OpenVMS Network Management Utilities	AA-PV61A-TK
DECnet for OpenVMS Networking Manual	AA-PV60A-TK
OpenVMS VAX Upgrade and Installation Supplement: VAX 8820, 8830, 8840	AA-PS6MA-TE
OpenVMS VAX Upgrade and Installation Supplement: VAX 8200, 8250, 8300, 8350	AA-PS6PA-TE
OpenVMS VAX Upgrade and Installation Supplement: VAX 8530, 8550, 8810 (8700), and 8820–N (8800)	AA-PS6QA-TE
OpenVMS VAX Upgrade and Installation Supplement: VAX 8600, 8650	AA-PS6UA-TE
VMS Upgrade and Installation Supplement: VAX-11/780, 785	AA-LB29B-TE
VMS Upgrade and Installation Supplement: VAX-11/750	AA-LB30B-TE

Descriptions of the archived OpenVMS manuals are as follows:

A Comparison of System Management on OpenVMS AXP and OpenVMS VAX

Discusses system management tools, the impact of Alpha page sizes on system management operations, the system directory structure, interoperability issues, and performance information. Designed for system managers who need to learn quickly how to manage an OpenVMS Alpha system.

Building Dependable Systems: The OpenVMS Approach

Offers practical information about analyzing the dependability requirements of your business applications and deciding how to use your computing systems to support your dependability goals. This information is complemented by technical summaries of the dependability features of OpenVMS and related hardware and layered software products.

Creating an OpenVMS Alpha Device Driver from an OpenVMS VAX Device Driver

Describes the procedures for converting a device driver used on OpenVMS VAX to a device driver that runs on OpenVMS Alpha. This book also contains data structures, routines, and macros for maintaining an Alpha driver written in Macro-32.

Creating an OpenVMS AXP Step 2 Device Driver from a Step 1 Device Driver

Provides information for upgrading a Step 1 device driver (used in earlier versions of OpenVMS AXP) to a Step 2 device driver. A Step 2 device driver is required for OpenVMS AXP Version 6.1.

Creating an OpenVMS AXP Step 2 Device Driver from an OpenVMS VAX Device Driver

Provides information for migrating a device driver used on OpenVMS VAX to a Step 2 device driver used on OpenVMS AXP Version 6.1.

Guide to OpenVMS AXP Performance Management

Introduces and explains the techniques used to optimize performance on an OpenVMS Alpha system.

Guide to OpenVMS Performance Management

Introduces and explains the techniques used to optimize performance on an OpenVMS VAX system.

Migrating an Application from OpenVMS VAX to OpenVMS Alpha

Describes how to create an OpenVMS Alpha version of an OpenVMS VAX application. 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, this manual 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.

Migrating an Environment from OpenVMS VAX to OpenVMS Alpha

Describes how to migrate a computing environment from an OpenVMS VAX system to an OpenVMS Alpha system or a mixed-architecture cluster. Provides an overview of the VAX to Alpha migration process and describes the differences in system and network management on VAX and Alpha computers.

Migrating to an OpenVMS AXP System: Planning for Migration

Describes the general characteristics of RISC architectures, compares the Alpha architecture to the VAX architecture, and presents an overview of the migration process and a summary of migration tools provided by HP. The information in this manual is intended to help you define the optimal migration strategy for your application.

Migrating to an OpenVMS AXP System: Recompiling and Relinking Applications

Provides detailed technical information for programmers who must migrate high-level language applications to OpenVMS Alpha. Describes how to set up a development environment to facilitate the migration of applications, helps programmers identify application dependencies on elements of the VAX architecture, and introduces compiler features that help resolve these dependencies. Individual sections of this manual discuss specific application dependencies on VAX architectural features, data porting issues (such as alignment concerns), and the process of migrating VAX shareable images.

OpenVMS Alpha Guide to 64-Bit Addressing and VLM Features

Introduces and describes OpenVMS Alpha operating system support for 64-bit virtual addressing and Very Large Memory (VLM). Intended for system and application programmers, this guide highlights the features and benefits of OpenVMS Alpha 64-bit and VLM capabilities. It also describes how to use these features to enhance application programs to support 64-bit addresses and to efficiently harness very large physical memory.

OpenVMS Alpha System Dump Analyzer Utility Manual

Explains how to use the System Dump Analyzer utility to investigate system failures and examine a running OpenVMS Alpha system. Alpha programmers should refer to this manual; VAX programmers should refer to the *OpenVMS VAX System Dump Analyzer Utility Manual*.

OpenVMS AXP Device Support: Developer's Guide

Describes how to write a driver for OpenVMS Alpha for a device not supplied by Compaq.

OpenVMS AXP Device Support: Reference

Provides the reference material for the *Writing OpenVMS Alpha Device Drivers in C* by describing the data structures, macros, and routines used in device-driver programming.

OpenVMS Bad Block Locator Utility Manual

Describes how to use the Bad Block Locator utility to locate bad blocks on older types of media.

OpenVMS Compatibility Between VAX and Alpha

Compares and contrasts OpenVMS on VAX and Alpha computers, focusing on the features provided to end users, system managers, and programmers.

OpenVMS Developer's Guide to VMSINSTAL

Describes the VMSINSTAL command procedure and provides guidelines for designing installation procedures that conform to standards recommended by Compaq. Intended for developers who are designing installation procedures for software products layered on the OpenVMS operating system.

OpenVMS DIGITAL Standard Runoff Reference Manual

Describes the DSR text-formatting utility.

OpenVMS EDT Reference Manual

Contains complete reference information for the EDT editor.

OpenVMS Exchange Utility Manual

Describes how to use the Exchange utility to transfer files between some foreign format volumes and OpenVMS native volumes.

OpenVMS Glossary

Defines terms specific to OpenVMS that are used throughout the documentation.

OpenVMS Guide to Extended File Specifications

Provides an overview of Extended File Specifications and describes the overall differences and impact Extended File Specifications introduce to the OpenVMS environment.

OpenVMS Master Index

Offers an edited compilation of indexes from the manuals in the OpenVMS Full Documentation set.

OpenVMS National Character Set Utility Manual

Describes how to use the National character set utility to build NCS definition files.

OpenVMS Obsolete Features Manual

Presents the DCL commands, system services, RTL routines, and utilities made obsolete by VMS Version 4.0 through Version 5.0. Includes an appendix of DCL commands, RTL routines, and utilities eliminated from VMS Version 4.0.

OpenVMS Programming Environment Manual

Provides a general description of Compaq products and tools that define the programming environment. Introduces facilities and tools such as the compilers, the linker, the debugger, the System Dump Analyzer, system services, and routine libraries.

OpenVMS Programming Interfaces: Calling a System Routine

Describes the OpenVMS programming interface and defines the standard conventions to call an OpenVMS system routine from a user procedure. The Alpha and VAX data type implementations for various high-level languages are also presented in this manual.

OpenVMS RTL DECtalk (DTK$) Manual

Documents the DECtalk support routines contained in the DTK$ facility of the OpenVMS Run-Time Library.

OpenVMS RTL Parallel Processing (PPL$) Manual

Documents the parallel-processing routines contained in the PPL$ facility of the OpenVMS Run-Time Library. Indicates which routines are specific to Alpha or VAX, as well as how routines function differently on each system.

OpenVMS Software Overview

Provides an overview of the OpenVMS operating system and some of its available products.

OpenVMS SUMSLP Utility Manual

Describes how to use the SUMSLP batch-oriented editor to update source files.

OpenVMS System Messages and Recovery Procedures Reference Manual

Contains an alphabetical listing of the errors, warnings, and informational messages issued by the operating system. Also provides the meaning of each message and a statement of the action to be taken in response to each message. This manual is in two volumes.

OpenVMS Terminal Fallback Utility Manual

Describes how to use the Terminal Fallback utility to manage the libraries, character conversion tables, and terminal parameters that are available within this utility.

OpenVMS VAX Card Reader, Line Printer, and LPA11–K I/O User's Reference Manual

Describes the card reader, laboratory peripheral accelerator, and line printer drivers on OpenVMS VAX.

OpenVMS VAX Device Support Manual

Describes how to write an OpenVMS VAX driver for a device not supplied by Compaq.

OpenVMS VAX Device Support Reference Manual

Provides the reference material for the *OpenVMS VAX Device Support Manual* by describing the data structures, macros, and routines used in device-driver programming.

OpenVMS VAX Patch Utility Manual

Describes how to use the Patch utility to examine and modify executable and shareable OpenVMS VAX images.

OpenVMS Wide Area Network I/O User's Reference Manual

Describes the DMC11/DMR11, DMP11 and DMF32, DR11-W and DRV11-WA, DR32, and asynchronous DDCMP interface drivers on OpenVMS VAX.

PDP–11 TECO User's Guide

Describes the operating procedures for the PDP-11 TECO (Text Editor and Corrector) program.

POLYCENTER Software Installation Utility User's Guide

Provides information on the POLYCENTER Software Installation utility, a new component that lets you install and manage software products that are compatible with the utility.

TCP/IP Networking on OpenVMS Systems

Provides an introductory overview of TCP/IP networking and describes OpenVMS DCL support for TCP/IP capabilities.

Index