HP OpenVMS Version 8.2–1 for Integrity Servers New Features and Release Notes

Order Number: BA322-90033

September 2005

This manual describes the new features and release notes for OpenVMS I64 Version 8.2–1.

Revision/Update Information:	This is a new manual.
Software Version:	OpenVMS Version 8.2–1 for Integrity Servers

Hewlett-Packard Company
Palo Alto, California

ZK6676

The HP OpenVMS documentation set is available on CD–ROM.

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface vii

1 Introduction

1.1 Overview of OpenVMS Version 8.2–1 for Integrity Servers 1–1
1.2 HP Software Technical Support Policy 1–2
1.3 General Application Compatibility Statement 1–3
1.4 Obtaining Patch Kits 1–3
1.5 Networking Options 1–3
1.6 Disk Incompatibility with Older Versions of OpenVMS 1–4
1.7 System Event Log (SEL) on Integrity Servers 1–4
1.8 Firmware for Integrity Servers 1–4
1.9 TIE Kit Must be Removed Before Upgrade 1–6
1.10 Release Notes on Booting the System 1–6
1.10.1 Booting from the Installation DVD 1–7
1.10.2 Setting Up I64 Systems to Reboot 1–7
1.10.3 Booting with a Common Cluster System Disk 1–7
1.10.4 Booting from a Fibre Channel Storage Device 1–8
1.10.5 OpenVMS I64 Boot Manager Utility: Adding Multipath Fibre Channel Disk Devices 1–8
1.10.6 Fibre Channel Boot Disk: Simplified Setup Process 1–8
1.11 HP DECwindows Motif 1–8
1.11.1 Connect Peripheral Devices Prior to Server Startup 1–9
1.11.2 Countdown Messages Displayed During Startup 1–9
1.11.3 VGA Console Not Supported for Booting 1–9
1.11.4 Optional Graphics 1–9
1.11.5 Keyboard Support 1–10

2 OpenVMS Version 8.2–1 New Features

2.1 New Integrity Server Support 2–1
2.1.1 HP sx1000 Chipset for HP Integrity Servers Supported Configurations 2–1
2.2 Booting the OpenVMS I64 Operating Environment DVD from the InfoServer 2–3
2.3 OpenVMS Cluster Systems 2–3
2.3.1 Increased Number of OpenVMS I64 Systems in an OpenVMS Cluster 2–3
2.3.2 Shared SCSI Storage Support for Two-Node OpenVMS I64 Cluster Systems 2–4
2.4 Ctrl/P Now Provides IPC Support on Integrity Server Consoles 2–5
2.5 HP OpenVMS Debugger 2–6
2.5.1 Improved C++ Support for Operator Names 2–6
2.5.2 Use of SET MODULE Command is Now Optional 2–6

2.5.3 Heap Analyzer Now Available . . . 2–6
2.5.4 New Qualifier for SHOW STACK Command . . . 2–6
2.5.5 Change to Default Data Type for Untyped Storage Locations . . . 2–7
2.5.6 Improved Overloaded Symbol Support in SHOW SYMBOL Command . . . 2–7
2.5.7 Preliminary Ada Language Support . . . 2–7
2.6 Debuggers Now Available . . . 2–7
2.7 Fibre Channel Performance Data with New SDA Command . . . 2–8
2.8 Larger Granularity Hints for Memory-Resident Sections . . . 2–10
2.8.1 Flags Added to MMG_CTLFLAGS System Parameter . . . 2–11
2.8.2 System Service Changes . . . 2–12
2.9 Partition Manager . . . 2–14
2.10 New Power-Saving Feature . . . 2–14
2.11 PPL Units Assignment Tool . . . 2–15
2.12 System Dump Analyzer (SDA) Utility . . . 2–15
2.12.1 SDA COPY Command Additional Information . . . 2–15
2.12.2 New Qualifiers for COPY Command . . . 2–16
2.12.3 New SDA Command . . . 2–16
COLLECT . . . 2–17
2.12.4 Changes to SHOW CALL_FRAME Parameter . . . 2–18
2.13 Traceback Facility . . . 2–18

3 OpenVMS V8.2–1 Release Notes

3.1 Adapter Release Notes . . . 3–1
3.1.1 A9782A, A9784A, and AB465A Restriction in Zoned SAN Environments . . . 3–1
3.1.2 Fibre Channel EFI Driver and Firmware Requirements . . . 3–1
3.2 Associated Product Support . . . 3–2
3.3 Cluster Compatibility Patch Kits . . . 3–2
3.4 HP OpenVMS Debugger Heap Analyzer Conditions and Workarounds . . . 3–4
3.5 Documentation Corrections . . . 3–4
3.5.1 *HP OpenVMS Debugger Manual*: Client/Server Interface Correction . . . 3–4
3.5.2 *HP OpenVMS I/O User's Reference Manual*: PTD$READ Clarification . . . 3–5
3.5.3 *HP OpenVMS Version 8.2 New Features and Documentation Overview*: Librarian Utility Corrections . . . 3–5
3.5.3.1 /REMOVE Qualifier Correction . . . 3–5
3.5.3.2 Accessing ELF Object Libraries Correction . . . 3–6
3.5.4 *HP OpenVMS RTL Library (LIB$) Manual* Corrections . . . 3–6
3.5.4.1 *HP OpenVMS RTL Library (LIB$) Manual*: Clarification of Rounding Rule for LIB$CVT_DX_DX . . . 3–7
3.5.5 *HP OpenVMS RTL Library (LIB$) Manual*: Clarification of Platform Restrictions . . . 3–7
3.5.6 *HP OpenVMS System Manager's Manual*: IPC Commands Restriction . . . 3–8
3.5.7 *HP OpenVMS System Services Reference Manual* Additions and Corrections . . . 3–8
3.5.7.1 $GETRMI Item Code—RMI$_MODES . . . 3–8
3.5.7.2 $PUTMSG System Service Correction . . . 3–10
3.5.8 *HP Volume Shadowing for OpenVMS*: Corrections to Memory Requirements . . . 3–10
3.6 EFI Shell Precautions on Shared or Shadowed System Disks . . . 3–10

3.7 Librarian: Increased Object Module Name Key Length 3–11
3.8 Linker Utility 3–12
3.9 Multiple nPartitions on Cell-based Systems 3–13
3.10 Network Update Restrictions from Version 8.2 to Version 8.2–1 3–13
3.11 NPAG_AGGRESSIVE and NPAG_GENTLE Default Values 3–13
3.12 Synchronous Data Links Not Supported 3–13
3.13 HP TCP/IP Services for OpenVMS 3–14
3.14 Traceback API Problem Fixed 3–14

4 InfoServer Utility

4.1 InfoServer Utility Overview 4–1
4.1.1 InfoServer Usage Summary 4–1
4.1.2 InfoServer Commands 4–2
CREATE SERVICE 4–3
DELETE SERVICE 4–7
EXIT 4–10
HELP 4–11
SAVE 4–12
SET SERVICE 4–15
SHOW SERVER 4–18
SHOW SERVICES 4–19
SHOW SESSIONS 4–21
SPAWN 4–23
START SERVER 4–24

5 Linker Utility

5.1 Linker Utility Overview 5–1
5.2 Differences When Linking on OpenVMS I64 Systems 5–2
5.2.1 Specifying Based Clusters Varies by OpenVMS Platform 5–2
5.2.2 Handling of Initialized Overlaid Program Sections on OpenVMS I64 Systems 5–3
5.2.3 Behavior Difference When Linking Relaxed Reference/Definition Symbols 5–6
5.2.4 Flags Set When /TRACEBACK, /DEBUG, and /DSF Are Used 5–6
5.3 New Aspects for Linking on OpenVMS I64 Systems 5–8
5.3.1 Understanding Linkage Messages 5–8
5.3.2 Considerations for Images Compiled with Reduced Floating–Point Model 5–11
5.3.3 Considerations for Linking with ELF Groups and UNIX-Style Weak Symbols 5–11
5.4 New Linker Qualifiers for OpenVMS I64 5–12
5.4.1 New /BASE_ADDRESS Qualifier 5–13
5.4.2 New /SEGMENT_ATTRIBUTE Qualifier 5–13
5.4.3 New /FP_MODE Qualifier 5–13
5.4.4 New Linker Qualifiers: /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS 5–14
5.4.5 New GROUP_SECTIONS and SECTION_DETAILS keywords for the /FULL Qualifier 5–17
5.5 Extensions to Linker Options for OpenVMS I64 5–17
5.5.1 New Alignments for the PSECT_ATTRIBUTE Option 5–17
5.6 New Ways to Use Existing Linker Qualifiers and Options 5–17

5.6.1 Mixed-Case Arguments in Linker Options on I64 Systems 5–18
5.6.2 Conventions for Specifying Image Names 5–18
5.6.3 Using the PSECT_ATTRIBUTE Option to Specify Alignment 5–19
5.6.4 Special Handling of Linker Nonexistent Files 5–19
5.7 New OpenVMS I64 Linker Map 5–21

6 Product Directories

6.1 OpenVMS I64 Operating Environment 6–1
6.1.1 Directories on the OpenVMS I64 Operating Environment DVD 6–1
6.2 Directories on the OpenVMS Freeware CD 6–3
6.3 Open Source Tools CD 6–3
6.3.1 Directories on the Open Source Tools CD 6–5
6.4 Product Licensing 6–5
6.5 OpenVMS Version 8.2–1 Documentation 6–6

Index

Examples

5–1 Additional information 5–4
5–2 Linker Map Showing Program Section Synopsis 5–4
5–3 Incompatible Initialization 5–5
5–4 Initialization Example on OpenVMS 5–5
5–5 Additional Incompatible Initialization 5–5

Figures

2–1 Two-Node OpenVMS I64 Cluster System 2–5
5–1 Object and Image Synopsis and Cluster Synopsis 5–22
5–2 Image Segment Synopsis 5–23
5–3 Program Section Synopsis 5–24
5–4 Program Section Synopsis (Continued) 5–25
5–5 Symbol Cross Reference 5–26
5–6 Symbols by Value 5–27
5–7 Image Synopsis 5–28
5–8 Link Run Statistics 5–29

Tables

1–1 Firmware Versions for Entry-Class Integrity Servers 1–5
2–1 FC Performance Command Qualifiers 2–8
3–1 Patch Kits Required for Cluster Compatibility 3–3
6–1 Directory Structure on the OpenVMS I64 Operating Environment DVD 6–2
6–2 OpenVMS Open Source Tools Version 3.0 CD 6–5

Preface

Intended Audience

This manual is intended for all users of the HP OpenVMS I64 Version 8.2–1 operating system. Read this manual before you install, upgrade, or use OpenVMS Version 8.2–1.

Document Structure

This manual contains the following chapters:

- Chapter 1 provides an overview of the release, discusses the HP technical support policy, and networking options.
- Chapter 2 describes the new features provided in OpenVMS Version 8.2–1 for Integrity servers.
- Chapter 3 contains release notes pertaining to Version 8.2–1. Notes are organized by facility or product name when applicable.
- Chapter 4 describes the InfoServer utility new feature, which allows you to create a service for a virtual disk device on the local area network.
- Chapter 5 provides the Linker utility overview as well as considerations you should review before linking programs on OpenVMS I64 systems.
- Chapter 6 provides the directory names and locations of the products on the Operating Environment (OE) DVD and other media included in the OpenVMS Version 8.2–1 media kit.

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 World Wide Web address:

`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.
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).
`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.
UPPERCASE TYPE	Uppercase type indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1
Introduction

This chapter contains an introduction to the HP OpenVMS Version 8.2–1 for Integrity servers product and information that you need to know before installing or upgrading to Version 8.2–1.

HP recommends that you read all of the following documents before installing or upgrading to OpenVMS Version 8.2–1:

- *HP OpenVMS Version 8.2–1 for Integrity Servers New Features and Release Notes* (this manual)
- *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*
- *Cover Letter for HP OpenVMS Version 8.2–1 for Integrity Servers*
- *HP OpenVMS Version 8.2 Release Notes*

1.1 Overview of OpenVMS Version 8.2–1 for Integrity Servers

OpenVMS Version 8.2–1 for Integrity servers (also known as OpenVMS I64) is a follow-on release that replaces OpenVMS Version 8.2 for Integrity servers. OpenVMS Version 8.2–1 includes all the capabilities of its predecessor and introduces new features and hardware support.

Major new features include:

- Support for the following Integrity server platforms:
  - — HP Integrity rx7620 server
  - — HP Integrity rx8620 server
  - — HP Integrity Superdome server

  The HP sx1000 Chipset for HP Integrity servers provides the CPU, memory and I/O subsystem for these servers.
- InfoServer utility

  Allows you to create a service for a virtual disk device on the local area network (LAN). Also allows you to boot from a virtual DVD drive on the LAN from which you can upgrade your system to Version 8.2–1.

  Supported on core LAN I/O devices on the rx1600, rx1620, rx2600, rx2620, and rx4640 Integrity servers.
- Support for nPartitions

  On HP cell-based servers (such as rx7620, rx8620, and Integrity Superdome) each **nPartition** defines a subset of the server hardware resources that is used as an independent system environment. A **cell-based server** is a hardware complex that can run one or more operating systems and that supports dividing hardware resources into nPartitions. In a cell-based server,

all processors and memory are contained in circuit boards called cells, each of which can be assigned for exclusive use by an nPartition. An nPartition has its own system boot interface, and each nPartition boots and reboots independently.

- Power-saver feature

  Minimizes power consumption by allowing you to place a CPU in low-power mode if idle, thus reducing energy costs.

- Low-cost two-node multihost SCSI clusters

  Shared SCSI storage support for two-node OpenVMS I64 Cluster systems, allowing affordable shared storage in a two-node cluster environment (entry-class Integrity servers only).

- Increase in the number of nodes in an I64 cluster

  Support of up to 96 nodes in an OpenVMS Cluster system, with no constraints on the mix of Integrity server and AlphaServer nodes.

- Fibre Channel performance data with new System Dump Analyzer command

  Fibre Channel performance data available with the new System Dump Analyzer command, FC PERFORMANCE.

- PPL units assignment tool

  A tool to help manage license units for OpenVMS Per Processor Licenses (PPL).

1.2 HP Software Technical Support Policy

Unless otherwise agreed to by Hewlett-Packard Company (HP), HP provides HP Services support only for the current and immediately preceding versions of HP software, and only when the software is used with hardware that is included in HP-specified configurations. A version is defined as a release of a software product that contains new features, enhancements, or maintenance updates.

Current version-level support (Standard Support, or SS) and Prior Version Support (PVS) for OpenVMS operating system software are provided for OpenVMS versions in accordance with these guidelines. The current level of support for recent versions of OpenVMS I64, as well as Alpha and OpenVMS VAX, is kept up to date on this Web site:

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

The following OpenVMS core products are supported at the same level (SS or PVS) and duration as the operating system version on which they ship:

- HP Advanced Server for OpenVMS
- HP DECnet (Phase IV)
- HP DECnet-Plus for OpenVMS
- HP OpenVMS Cluster Client Software
- HP OpenVMS Cluster Software for OpenVMS
- HP RMS Journaling for OpenVMS
- HP TCP/IP Services for OpenVMS
- HP Volume Shadowing for OpenVMS

These products require their own individual support contracts and are not included in the operating system support contract.

1.3 General Application Compatibility Statement

OpenVMS has consistently held the policy that published APIs are supported on all subsequent releases. It is unlikely that applications that use published APIs will require changes to support a new release of OpenVMS. APIs may be "retired" and thus be removed from the documentation; however, the API continues to be available on OpenVMS as an undocumented interface.

1.4 Obtaining Patch Kits

Patch kits, also known as remedial kits, for HP products are available on line at the HP IT Resource Center (ITRC). Use of the ITRC patch download site requires user registration and login. Registration is open to all users and no service contract is required. You can register and log in from the following URL:

`http://www2.itrc.hp.com/service/patch/mainPage.do`

You can also use FTP to access patches from the following location:

`ftp://ftp.itrc.hp.com/openvms_patches`

1.5 Networking Options

OpenVMS provides customers with the flexibility to choose their own network protocol. Whether you require DECnet or TCP/IP, OpenVMS allows you to choose the protocol or combination of protocols that work best for your network. OpenVMS can operate with both HP and third-party networking products.

During the main installation procedure for OpenVMS Version 8.2–1, you have the option of installing the following supported HP networking software:

- Either HP DECnet-Plus Version 8.2–1 for OpenVMS or HP DECnet Phase IV Version 8.2–1 for OpenVMS. (Note that these DECnet products cannot run concurrently on your system.)

  DECnet-Plus contains all the functionality of the DECnet Phase IV product, plus the ability to run DECnet over TCP/IP or OSI protocols.

  Support for DECnet Phase IV is provided to customers with a Prior Version Support Contract. For more information about the Prior Version Support service, refer to Section 1.2.

- HP TCP/IP Services for OpenVMS Version 5.5

  TCP/IP Services and DECnet can run concurrently on your system. Once you have installed HP DECnet-Plus for OpenVMS and TCP/IP Services on your system, you can run DECnet applications and OSI applications, or both, over your TCP/IP network. Refer to the *DECnet-Plus for OpenVMS Management Guide* for more information about running DECnet over TCP/IP (RFC 1859) and OSI over TCP/IP (RFC 1006).

Alternatively, after you install OpenVMS, you can install your choice of another third-party networking product that runs on OpenVMS Version 8.2–1.

For information about how to configure and manage your HP networking software after installation, refer to the TCP/IP, DECnet-Plus, or DECnet documentation. The manuals are available in online format on the OpenVMS

Documentation CD–ROM, the Online Documentation Library CD, and the OpenVMS Documentation Web site:

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

To order printed manuals from HP, call 800-282-6672.

1.6 Disk Incompatibility with Older Versions of OpenVMS

The OpenVMS Version 8.2–1 installation procedure initializes the target disk with volume expansion (INITIALIZE/LIMIT). This renders the disk incompatible with versions of OpenVMS prior to Version 7.2. If you intend to mount this new disk on a version of OpenVMS prior to Version 7.2, you must first take steps to make the disk compatible for that operating system version. Refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual* for detailed instructions.

Note that by taking these steps, your new system disk might include a relatively large minimum allocation size (as defined by /CLUSTER_SIZE). As a result, small files will use more space than necessary. Therefore, you should perform these steps *only* for system disks that must be mounted on versions of OpenVMS prior to Version 7.2.

Note

ODS-5 disks are also incompatible with versions of OpenVMS prior to Version 7.2.

1.7 System Event Log (SEL) on Integrity Servers

HP Integrity servers maintain a System Event Log (SEL) within system console storage, and OpenVMS I64 automatically transfers the contents of the SEL into the OpenVMS error log. If you are operating from the console during a successful boot operation, you might see a message indicating that the Baseboard Management Controller (BMC) SEL is full. You can safely continue when the BMC SEL is full by following the prompts; OpenVMS automatically processes and clears the contents of the SEL.

1.8 Firmware for Integrity Servers

OpenVMS I64 Version 8.2–1 was tested with the latest firmware for each of the supported Integrity servers.

For the entry-class Integrity servers, HP recommends that use the most current system firmware. For information about updating the system firmware for entry-class Integrity servers, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*. For rx7620, rx8620, and Superdome servers, call HP Customer Support to update your firmware.

The following table lists the recommended firmware versions for entry-class Integrity servers:

Table 1–1 Firmware Versions for Entry-Class Integrity Servers

System	System Firmware	BMC Firmware	MP Firmware	DHCP Firmware
rx1600	2.18	3.48	E.03.15	N/A
rx1620	2.18	3.48	E.03.15	N/A
rx2600	2.31	1.53	E.03.15	N/A
rx2620	3.17	3.47	E.03.15	N/A
rx4640	3.17	3.49	E.03.15	1.10

For cell-based servers, you must access the MP Command Menu and issue the *sysrev* command to list the MP firmware revision level. The *sysrev* command is available on all HP Integrity servers that have an MP. Note the EFI *info fw* command does not display the Management Processor (MP) firmware version on cell-based Integrity servers.

To check firmware version information on an entry-class Integrity server that does not have the MP, use the `info fw` command at the EFI prompt. Note the following example:

```
Shell> info fw

FIRMWARE INFORMATION

   Firmware Revision: 2.13 [4412]          ❶

   PAL_A Revision: 7.31/5.37
   PAL_B Revision: 5.65
   HI Revision: 1.02

   SAL Spec Revision: 3.01
   SAL_A Revision: 2.00
   SAL_B Revision: 2.13

   EFI Spec Revision: 1.10
   EFI Intel Drop Revision: 14.61
   EFI Build Revision: 2.10

   POSSE Revision: 0.10

   ACPI Revision: 7.00

   BMC Revision: 2.35                      ❷
   IPMI Revision: 1.00
   SMBIOS Revision: 2.3.2a
   Management Processor Revision: E.02.29  ❸
```

❶ The system firmware revision is 2.13.

❷ The BMC firmware revision is 2.35.

❸ The MP firmware revision is E.02.29.

Note

The versions displayed in this example might not reflect the versions running on your system.

The HP Integrity rx4640 server contains Dual Hot Plug Controller (DHPC) hardware with upgradable firmware. To check the current version of your DHPC firmware, enter the EFI command *info chiprev*, as shown in the following example. The hot-plug controller version will be displayed. A display of 0100 indicates version 1.0; a display of 0110 indicates version 1.1.

```
Shell> info chiprev

CHIP REVISION INFORMATION

   Chip                  Logical     Device       Chip
   Type                     ID         ID       Revision
   -------------------   -------     ------     --------
   Memory Controller         0       122b         0023
   Root Bridge               0       1229         0023
     Host Bridge          0000       122e         0032
     Host Bridge          0001       122e         0032
     Host Bridge          0002       122e         0032
     Host Bridge          0004       122e         0032
      HotPlug Controller     0          0         0110
     Host Bridge          0005       122e         0032
      HotPlug Controller     0          0         0110
     Host Bridge          0006       122e         0032
     Other Bridge              0          0         0002
       Other Bridge            0          0         0008
         Baseboard MC          0          0         0235
```

For instructions on how to access and use EFI, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*. For more information, refer to the hardware documentation that is provided with your server.

For instructions on upgrading your firmware for your entry-class Integrity servers, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*. To upgrade firmware for the rx7620, rx8620, or Superdome, contact HP Customer Support.

1.9 TIE Kit Must be Removed Before Upgrade

The Translated Image Environment (TIE) has been integrated into OpenVMS I64 Version 8.2–1. Refer to the HP OpenVMS Systems Migration Software Web site for further information.

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

If you have installed the Translated Image Environment (TIE) PCSI kit (HP-I64VMS-TIE) on OpenVMS I64 Version 8.2, you must manually remove any previous versions of TIE before you upgrade to OpenVMS I64 Version 8.2–1.

Use the following command to remove the TIE product kit:

```
$ PRODUCT REMOVE TIE
```

Do not install the TIE product kit, HP I64VMS TIE V1.0, on OpenVMS I64 Version 8.2–1.

1.10 Release Notes on Booting the System

The following release notes pertain to booting the OpenVMS I64 system.

1.10.1 Booting from the Installation DVD

On I64 systems with the minimum amount of supported memory, the following message appears when booting from the installation DVD:

```
********* XFC-W-MemmgtInit Misconfigure Detected ********
XFC-E-MemMisconfigure MPW_HILIM + FREEGOAL > Physical Memory and no reserved memory for XFC
XFC-I-RECONFIG Setting MPW$GL_HILIM to no more than 25% of physical memory XFC-I-RECONFIG
Setting FREEGOAL to no more than 10% of physical memory
********* XFC-W-MemMisconfigure AUTOGEN should be run to correct configuration ********
********* XFC-I-MemmgtInit Bootstrap continuing ********
```

The message means that the system cache (XFC) initialization has successfully adjusted the SYSGEN parameters MPW_HILIM and FREEGOAL to allow caching to be effective during the installation. You can continue with the installation.

1.10.2 Setting Up I64 Systems to Reboot

An OpenVMS I64 system does not reboot automatically unless you have it set up to do so either by using EFI or by using the OpenVMS I64 Boot Manager utility. In a cluster, the Alpha systems reboot but the I64 systems shut down and sit at the console prompt.

For information about how to set up your I64 system to reboot, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*.

1.10.3 Booting with a Common Cluster System Disk

For configuring additional nodes to boot with a common cluster disk on an OpenVMS Cluster system, HP requires that you execute the OpenVMS I64 Boot Manager (BOOT_OPTIONS.COM).

After you have enabled clustering on a single or standalone system, you can add additional I64 nodes to boot on a shared disk, as follows:

1. Boot the HP OpenVMS Version 8.2–1 Installation Disk on the target node.
2. From the OpenVMS Installation Menu; choose Option 7, Execute DCL commands and procedures, to access the DCL prompt.
3. At the DCL prompt, enter the following command to invoke the OpenVMS I64 Boot Manager utility:

   ```
   $$$ @$SYS$MANAGER:BOOT_OPTIONS
   ```

4. When the utility is invoked, the main menu is displayed. To add a system disk as a boot option, enter 1 (1) ADD an entry to the Boot Options list.
5. The utility will continue to prompt you for additional information, such as the device name, position in the EFI boot option list, boot flags setting, and boot option description.
6. When you have added your boot option, exit from the utility by entering E at the prompt.
7. Log out from the DCL prompt and shut down the I64 system.

For additional information on the I64 Boot Manager Boot Options Management Utility, refer to the *HP OpenVMS System Manager's Manual, Volume 1: Essentials*.

1.10.4 Booting from a Fibre Channel Storage Device

Many customers prefer to boot from a Fibre Channel (FC) storage device because of its speed and because it can serve as a common cluster system disk in a SAN. Booting from an FC storage device on OpenVMS I64 systems is significantly different from booting from an FC storage device on OpenVMS Alpha systems.

Refer to the Fibre Channel chapter of *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual* for directions on how to configure and boot from an FC device on OpenVMS I64 systems.

1.10.5 OpenVMS I64 Boot Manager Utility: Adding Multipath Fibre Channel Disk Devices

The OpenVMS Boot Manager utility, BOOT_OPTIONS.COM, is used to specify a list of boot and dump devices in a SAN. When you add a multipath Fibre Channel disk device to the list, all paths to the device found in the SAN, including redundant paths, are listed. You can delete redundant paths from the list by using the Remove option.

1.10.6 Fibre Channel Boot Disk: Simplified Setup Process

For OpenVMS I64 Version 8.2, the process of setting up a Fibre Channel boot device required the use of the OpenVMS I64 Boot Manager, BOOT_ OPTIONS.COM, to specify values to the EFI Boot Manager. This process has been automated by the OpenVMS I64 Version 8.2–1 installation process, although the manual process is still available for those cases when it might be needed.

The OpenVMS I64 Version 8.2–1 installation process displays the name of a Fibre Channel disk as a boot device and prompts you to add the Boot Option. HP recommends that you accept this default. Alternatively, you can run the OpenVMS I64 Boot Manager after the installation or upgrade completes, as described in the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*.

Note

If your system is a member of the rx1600, rx2600, or rx4600 family of servers and a Fibre Channel boot device is not listed in the EFI boot menu, you might experience a delay in the EFI initialization because the entire SAN is scanned.

Depending on the size of the SAN, this delay can range from several seconds to several minutes. Cell-based systems (the rx7620, rx8620, and Superdome families of servers) are not affected by this delay. This delay might be experienced when booting OpenVMS from the installation DVD for the first time on any OpenVMS I64 system.

Refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual* for information on booting from a Fibre Channel boot device and updating the Fibre Channel adapter firmware.

1.11 HP DECwindows Motif

The following DECwindows Motif release notes are of interest to OpenVMS I64 users.

1.11.1 Connect Peripheral Devices Prior to Server Startup

To properly configure your system as a DECwindows X display server, you must have all the following peripheral components connected prior to startup:

- Monitor
- USB mouse
- USB keyboard

Otherwise, the server system might not complete the device initialization process correctly. For example, starting up a server system without input devices (mouse and keyboard) will result in a blank screen, and DECwindows will not start .

To correct this problem, connect all peripherals. Then restart the server, or reboot the system.

1.11.2 Countdown Messages Displayed During Startup

When running DECwindows Motif in client-only mode (with no server configured, or with no mouse or keyboard connected), messages similar to the following might be displayed during startup:

```
Waiting for mouse...
Waiting for keyboard...
```

These messages indicate that device polling is underway and are informational only. They will disappear when the 15-second countdown is complete. This typically occurs on servers that incorporate built-in graphics as part of a Server Management Option, and no keyboard and mouse are connected.

To prevent the messages from displaying, and the 15-second delay, connect the input devices (USB mouse and USB keyboard) to the system prior to startup.

If you do not intend to use local graphics on the system, you can define the logical name in SYSTARTUP_VMS.COM as follows:

```
$ DEFINE/SYSTEM DECW$IGNORE_WORKSTATION TRUE
```

This prevents the DECwindows startup from attempting to start local graphics operation.

1.11.3 VGA Console Not Supported for Booting

The VGA Text Console is not supported by OpenVMS for booting. The Integrity server firmware allows input from the VGA, but following a BOOT command, an error message appears indicating that the device cannot be used. The boot continues with console output directed to a serial port (or the Server Management Option serial console port when connected). DECwindows starts and runs on the graphics display normally. Interactive boot support requires a terminal or a system running a terminal emulator connected to a serial port on the system. Other options, such as a LAN connected console, are also available.

1.11.4 Optional Graphics

The ATI Radeon 7500 PCI option (HP part number AB551A) is supported on the entry-class Integrity servers for 2D multi-head and 3D operation. Refer to the Installation Guide for this device or Quickspecs for information about configuration and use.

1.11.5 Keyboard Support

The OpenVMS USB keyboard (HP part number AB552A) is supported on all Integrity systems supporting DECwindows graphics. The keyboard is packaged with a 3-button thumbwheel mouse.

2

OpenVMS Version 8.2–1 New Features

This chapter describes the new features provided in OpenVMS I64 Version 8.2–1. In addition, Chapter 4 describes the InfoServer utility features and Chapter 5 describes the Linker utility features.

2.1 New Integrity Server Support

Starting with Version 8.2–1, OpenVMS I64 supports three more Integrity server platforms:

- HP Integrity rx7620 server

  Designed to operate as a single 2-way to 8-way (Intel® Itanium® 2 processor) SMP server or to divide it into two independent hard partitions (nPartitions).

- HP Integrity rx8620 server

  Builds on the rx7620 server with the addition of a crossbar backplane and two more cell boards. Operates as a 2-way to 16-way (Intel® Itanium® 2 processor) SMP server or can be divided into smaller independent nPartitions.

- HP Integrity Superdome server

  High-end Integrity server that includes the following models:

  – SD16A (supports up to 4 cells in a server complex)

  – SD32A (supports up to 8 cells in a server complex)

  – SD64A (supports up to 16 cells in a server complex)

2.1.1 HP sx1000 Chipset for HP Integrity Servers Supported Configurations

The HP sx1000 Chipset for HP Integrity servers provides the CPU, memory, and I/O subsystem to the HP Integrity rx7620, HP Integrity 8620, and HP Integrity Superdome servers. The cell controller is combined with four CPU chips into the computing cell in the sx1000 Chipset architecture. The cell controller chip also provides paths to the I/O devices and off-cell memory.

The rx7620, rx8620, and Superdome servers provide a varying number of sx1000 Chipset cells. The rx7620 provides up to 2 cells (8 CPUs), the rx8620 provides up to 4 cells (16 CPUs), and the Superdome provides up to 16 cells (64 CPUs).

OpenVMS I64 Version 8.2–1 supports the following configurations on the rx7620, rx8620, and the Superdome:

- CPU—Each cell contains up to 4 CPUs. OpenVMS supports the following revisions of CPUs:

  — rx7620

    * Madison 9, 1.6 GHz, 6 MB cache CPUs

    * Madison 9, 1.5 GHz, 4 MB cache CPUs

  - — rx8620
    - * Madison 9, 1.6 GHz, 6 MB cache CPUs
    - * Madison 9, 1.5 GHz, 4 MB cache CPUs
  - — Superdome—Madison 9, 1.6 GHz, 9 MB cache CPUs
- Memory
  - — Maximum memory per cell for rx7620 (16 DIMM slots per cell):
    - * 2 GB DIMMs, 32 GB per cell, 64 GB maximum per system (2-cell boards)
    - * 4 GB DIMMs, 64 GB per cell, 128 GB maximum per system
  - — Maximum memory per cell for rx8620 (16 DIMM slots per cell):
    - * 2 GB DIMMs, 32 GB per cell, 128 GB maximum per system (4-cell boards)
    - * 4 GB DIMMs, 64 GB per cell, 256 GB maximum per system
  - — Maximum memory per cell for Superdome (32 DIMM slots per cell, maximum 4 cells per nPartition):
    - * 1 GB DIMMs, 32 GB per cell, 128 GB per 4-cell nPartition
    - * 2 GB DIMMs, 64 GB per cell, 256 GB per 4-cell nPartition
- Core I/O
  - — rx7620—1 or 2 Core I/O boards, each containing an Ultra 160 LVD SCSI port, a 10/100/1000Base-T LAN port, 2 RS-232 serial ports (1 console and 1 uninterruptible power supply (UPS)), and a 10/100Base-T port (web and LAN console connections).
  - — rx8620—1 to 4 Core I/O boards (in two system expansion unit cabinets), each containing an Ultra3 LVD SCSI port, a 10/100/1000Base-T LAN port, 2 RS-232 serial ports (1 console and 1 UPS), and a 10/100Base-T port (Web and LAN console connections).
  - — Superdome—1 to 16 Core I/O boards (a general-purpose 10/100/Base-T LAN card), each in slot 0 of the PCI chassis. Multiple Core I/O can be present within a partition; however, only one can be active at a time.
- PCI Chassis
  - — MAX PCI I/O: rx7620 and rx8620, 8 slots per Core I/O (cell board must be in place)
  - — MAX PCI I/O: Superdome, 12 slots per cell (slot 0 must have the Core I/O board if this is a primary cell in a partition)
- Hard partitions
  - — rx7620—1 to 2 nPartitions (2 nPartitions represents 1 for each cell present on the system.)
  - — rx8620—1 to 4 nPartitions (4 nPartitions represents 1 for each cell present on the system.) A system expansion unit (SEU) is required for more than 2 nPartitions.

— Superdome—4 to 16 nPartitions (16 nPartitions represents 1 for each cell present on the system.) The maximum number of cells allowed in a Superdome nPartition with OpenVMS Version 8.2–1 is 4 (up to 16 CPUs).

2.2 Booting the OpenVMS I64 Operating Environment DVD from the InfoServer

OpenVMS I64 Version 8.2–1 supports booting from a virtual DVD drive on the local area network (LAN) using InfoServer software available with your OpenVMS Version 8.2–1 operating system. This feature is available only for operating system upgrades and only with the following Integrity servers:

- rx1600
- rx1620
- rx2600
- rx2620
- rx4640

The InfoServer software provides the additional advantage of allowing you to boot multiple systems on the network from a single copy of the distribution kit.

Network booting using the InfoServer requires several configuration steps unique to OpenVMS V8.2–1. Any configuration performed for network booting from an OpenVMS Alpha system using the InfoServer hardware will not be valid for OpenVMS I64 systems. For information on enabling your system to boot over the LAN using the OpenVMS I64 InfoServer software, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*. For more information about the InfoServer utility, refer to Section 4.1.

2.3 OpenVMS Cluster Systems

This section discusses several new OpenVMS Cluster features that are available in this release.

2.3.1 Increased Number of OpenVMS I64 Systems in an OpenVMS Cluster

Starting with Version 8.2–1, OpenVMS supports 96 OpenVMS systems in a mixed-architecture OpenVMS Cluster system consisting of OpenVMS I64 systems and OpenVMS Alpha systems, or in a single-architecture cluster of OpenVMS I64 systems only. Prior to this release, the total number of OpenVMS I64 systems supported in any type of OpenVMS Cluster system was 8. In mixed-architecture clusters of OpenVMS I64 and OpenVMS Alpha systems, a total of 16 systems was supported.

Customers with large OpenVMS Cluster environments can now do the following:

- Add more systems (either I64 or Alpha) to mixed-architecture OpenVMS Cluster systems that were formerly restricted to 16 members.
- Add OpenVMS I64 systems to Alpha-only clusters that already contain 16 or more systems.

For more detailed information about OpenVMS Cluster support, refer to *Guidelines for OpenVMS Cluster Configurations* and to the *HP OpenVMS Cluster Systems* manual.

2.3.2 Shared SCSI Storage Support for Two-Node OpenVMS I64 Cluster Systems

Shared SCSI storage support for two-node OpenVMS I64 Cluster systems is introduced in OpenVMS Version 8.2–1. Prior to this release, shared SCSI storage was supported on OpenVMS Alpha systems, only, using earlier SCSI host-based adapters (HBAs)

Shared SCSI storage in an OpenVMS Cluster system enables computers connected to a single SCSI bus to share access to SCSI storage devices directly. This capability makes it possible to build highly available servers using shared access to SCSI storage.

Shared SCSI storage in an OpenVMS I64 Cluster system is subject to the following restrictions:

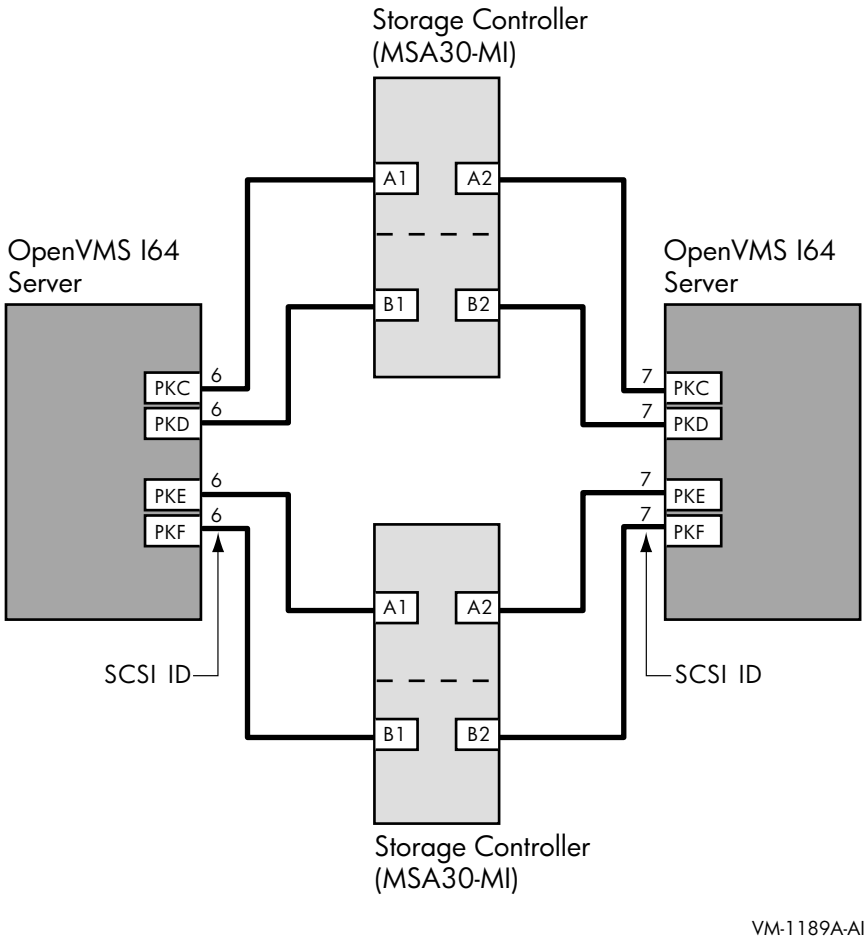
- A maximum of two OpenVMS I64 systems can be connected to a single SCSI bus.
- A maximum of four shared-SCSI buses can be connected to each system.
- Systems supported are the rx1600 family, the rx2600 family, and the rx4640.
- The A7173A HBA is the only supported HBA.
- MSA30-MI storage enclosure is the only supported SCSI storage type.
- Ultra320 SCSI disk family is the only supported disk family.

Figure 2–1 illustrates this configuration. Note that a second interconnect, a local area network (LAN), is required for host-to-host OpenVMS Cluster communications. (OpenVMS Cluster communications are also known as SCA (System Communications Architecture) communications.)

Note, too, the SCSI IDs of 6 and 7, which are required in this configuration. One of the systems must have a SCSI ID of 6 for each A7173A adapter port connected to a shared SCSI bus, instead of the factory-set default of 7. You use the U320_SCSI pscsi.efi utility, included on the IPF Offline Diagnostics and Utilities CD, to change the SCSI ID. The procedure for doing this is documented in the *HP A7173A PCI-X Dual Channel Ultra320 SCSI Host Bus Adapter Installation Guide*, which is available (PDF and HTML) from:

`http://docs.hp.com/en/netcom.html`

Figure 2–1 Two-Node OpenVMS I64 Cluster System



2.4 Ctrl/P Now Provides IPC Support on Integrity Server Consoles

With OpenVMS Version 8.2, pressing Ctrl/P on the console of an Integrity server displayed a prompt asking whether you wanted to crash the system:

```
Crash (y/n):
```

However, if the XDELTA debugger was loaded on your system, pressing Ctrl/P invokes the debugger. With OpenVMS Version 8.2–1, the behavior of Ctrl/P on the console has changed to invoke the IPC utility instead of the crash prompt. If XDELTA is loaded, Ctrl/P still results in transferring control to the XDELTA debugger. The Ctrl/P utility allows a system manager to adjust quorum or cancel mount verification on a disk.

The IPC utility has been updated with the following changes:

- A small help screen is displayed.
- If IPC is invoked and no input is typed within 45 seconds, the utility exits.
- If IPC fails to respond to Ctrl/P, pressing Ctrl/P again on the console may display the `Crash (y/n):` prompt.

- Pressing a Ctrl/P from within IPC displays the `Crash (y/n):` prompt. There is no time-out period for responding to this prompt.

The IPC utility runs at a high-interrupt priority level on the primary CPU. The utility still allows cluster communications to occur, but HP recommends that the utility be used and exited as soon as possible.

2.5 HP OpenVMS Debugger

The following sections describe new features of the OpenVMS Debugger on OpenVMS V8.2–1 Integrity systems.

2.5.1 Improved C++ Support for Operator Names

Support has been improved for C++ operator names. In particular, user-defined operator names are now supported. Prior to this change, you needed 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==
```

2.5.2 Use of SET MODULE Command is Now Optional

The OpenVMS Debugger can now set modules automatically, making use of an explicit SET MODULE command optional. There are two reasons for this enhancement:

- 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.

2.5.3 Heap Analyzer Now Available

The Heap Analyzer is now available on OpenVMS I64 and is activated differently than on Alpha systems. A new start command, START HEAP_ANALYZER, is available from the debugger command-line prompt (DBG>) to start the Heap Analyzer from the OpenVMS I64 debugger. For complete information, refer to the Heap Analyzer chapter in the *HP OpenVMS Debugger Manual*.

2.5.4 New Qualifier for SHOW STACK Command

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
```

2.5.5 Change to Default Data Type for Untyped Storage Locations

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.

2.5.6 Improved Overloaded Symbol Support in SHOW SYMBOL Command

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
```

2.5.7 Preliminary Ada Language Support

The OpenVMS I64 debugger has been enhanced to support the Ada language. This support is preliminary and will be enhanced in future releases of OpenVMS. Currently, the following capabilities are supported:

- Simple types, arrays and records
- Biased types
- Encapsulated types
- Packages
- Overloaded routine and data symbols
- User-defined operators

2.6 Debuggers Now Available

The following debuggers are included in the OpenVMS Version 8.2–1 Integrity server operating system:

- The HP DELTA debugger is now available on OpenVMS I64 systems.

  To use the DELTA debugger on OpenVMS I64, follow the directions for running DELTA on OpenVMS Alpha systems. The commands accepted by the DELTA debugger on OpenVMS I64 are the same as on OpenVMS Alpha. However, the register names recognized by DELTA on OpenVMS I64 are the Intel® Itanium® register names, the same ones accepted by XDELTA on OpenVMS I64.

- System Code Debugger (SCD)

  SCD allows you to debug nonpageable system code and device drivers running at any interrupt priority level (IPL).

- System Dump Debugger (SDD)

  SDD allows you to analyze certain system dumps using the commands and semantics of SCD.

2.7 Fibre Channel Performance Data with New SDA Command

A new System Dump Analyzer (SDA) command, FC PERFORMANCE, is available in OpenVMS Version 8.2–1 to display I/O performance characteristics of DGA devices. It is also available in OpenVMS Version 8.2 of OpenVMS Alpha and OpenVMS I64. In addition, this command is available for OpenVMS Alpha Version 7.3-2 in the FIBRE_SCSI_V400 Fibre Channel patch kit. The Fibre Channel drivers of these supported versions keep a performance array for every configured disk.

You can use this new SDA command to display I/O performance characteristics of a named DGA device or of all DGA devices that are configured on the system. If you omit the device name, then the performance data of all DGA disks with any nonzero data is displayed, regardless of whether the devices are currently mounted. FC PERFORMANCE arrays keep counts of I/O latency across LUNs, I/O size, and I/O direction (read/write).

I/O latency is measured from the time a request is queued to the host FC adapter until the completion interrupt occurs. Modern disk drives have an average latency in the range of 5-10 ms. Caches located in disk controllers (HSG/EVA /MSA/XP) and in the physical disk drives can occasionally produce significantly lower access times.

By default, the FC PERFORMANCE command uses the /SYSTIME qualifier to measure latency. The /SYSTIME qualifier uses EXE$GQ_SYSTIME, which is updated every millisecond. If I/O completes in less than 1 ms, it appears to have completed in zero time. When /SYSTIME is used, I/O operations that complete in less than 1 ms are shown in the display in the <2us column, where us represents microsecond.

To achieve greater accuracy, you can use the /RSCC qualifier, which uses the System Cycle Counter timer. The command qualifiers, including the timers, are described in Table 2–1.

Table 2–1 FC Performance Command Qualifiers

Qualifier	Description
[*device-name*]	Device whose performance characteristics are to be displayed. You can specify only one DGA device name. If you omit the name, then the performance data for every DGA device configured on the system is displayed, provided the array contains nonzero data. This includes the performance data for DGA devices that are not currently mounted.
[/RSCC | /SYSTIME]	Two time qualifiers are available. The /RSCC qualifier uses a PAL call, Read System Cycle Counter, to get a CPU cycle counter. This is highly accurate but incurs the cost, in time, of two expensive PAL calls per I/O operation. To display time resolution below 1 ms, use this qualifier. The /SYSTIME qualifier is the OpenVMS system time which is updated every millisecond; it is the default.

(continued on next page)

Table 2–1 (Cont.) FC Performance Command Qualifiers

Qualifier	Description
[/COMPRESS]	Suppresses the screen display of columns that contain only zeroes.
[/CSV]	Causes output to be written to a comma-separated values (CSV) file, which can be read into a Microsoft® Excel spreadsheet or can be graphed.
[/CLEAR]	Clears the performance array. This qualifier is useful when you are testing I/O performance on your system. Before starting a new test, you can clear the performance array for the device whose performance you are measuring and then immediately dump the contents of its performance array when the test is completed.

The following example shows the write and read output for device 1DGA4321 resulting from the use of the SDA command FC PERFORMANCE with the /COMPRESS qualifier.

In the display, LBC (first column heading) represents logical block count. Notice that the LBCs in the rows are powers of 2. A particular row contains all counts up to the count in the next row. A particular row contains all LBC counts up to the count that begins the next row; for example, the LBC 8 row shows the count for all I/Os with LBC 8 through LCB 15. I/Os with an LBC greater than 256 are not shown in the matrix, but they are included in the "total blocks" value at the beginning of the matrix.

The counts in each column represent the number of I/Os that completed in less than the time shown in the column header. For example the <2ms column means that the I/Os in this column took less than 2ms but more than 1 ms to complete. Similarly, the I/Os in the <4ms column took less than 4ms but more than 2 ms to complete. The one exception is the <2us column when you use the /SYSTIME qualifier; all I/O that completes in less than a millisecond are included in the <2us count.

The columns headings of <2us, <2ms, <4ms, and so on, are shown in this display. If there are no values for some of the headings, those columns are not displayed because the /COMPRESS qualifier was used. If the /RSCC qualifier was used instead of the default /SYSTIME qualifier, additional headings for <4us, <8us, <16us, and <256us would be displayed.

```
SDA> FC PERFORMANCE $1$DGA4321/COMPRESS

Fibre Channel Disk Performance Data
-----------------------------------

$1$dga4321 (write)

Using EXE$GQ_SYSTIME to calculate the I/O time

    accumulated write time = 2907297312us

    writes = 266709

    total blocks = 1432966

I/O rate is less than 1 mb/sec

LBC  <2us  <2ms  <4ms  <8ms <16ms <32ms <64ms <128ms <256ms <512ms <1s

===  ===== ====  ====  ==== ===== ===== ===== ====   =====  ===== ====
```

```
  1   46106 20630 12396 13605 13856 15334 14675 8101    777     8    -   145488
  2      52    21     8     9     5     5     6    1      2     -    -      109
  4   40310 13166  3241  3545  3423  3116  2351  977     88     -    -    70217
  8    2213  1355   360   264   205   225   164   82      5     -    -     4873
 16   16202  6897  3283  3553  3184  2863  2323 1012    108     -    1    39426
 32     678   310    36    39    47    44    33   27      6     -    -     1220
 64     105    97    18    26    41    43    42   24      7     -    -      403
128     592  3642   555    60    43    31    23    9      2     -    -     4957
256       -     9     7     -     -     -     -    -      -     -    -       16

     106258 46127 19904  21101 20804 21661 19617 10233  995     8    1   266709

Fibre Channel Disk Performance Data
----------------------------------
$1$dga4321 (read)
Using EXE$GQ_SYSTIME to calculate the I/O time
    accumulated read time = 1241806687us
    reads = 358490
    total blocks = 1110830

I/O rate is less than 1 mb/sec

LBC  <2us  <2ms  <4ms  <8ms  <16ms <32ms <64ms <128ms <256ms <512ms <2s
=== ===== =====  ===== ===== ===== =====  ===== ===== ====== ====== ====
  1  46620 12755  6587  7767  3758  2643   1133   198     5    -    -    81466
  2    574   134    66   158    82    20     21     4     1    -    -     1060
  4 162060 35896 20059 18677 15851 11298   5527  1300    25    2    1   270696
  8    355    79    46    97    59    36     28    10     -    -    -      710
 16    241   103    32   150    77    24     13     1     -    -    -      641
 32    916   355    76   302   316    61     25    10     -    -    -     2061
 64    725   380    64   248   140    17     10     3     -    -    -     1587
128     13    22    13    36    21     6      -     -     -    -    -      111
256     10    41    28    15    49    13      2     -     -    -    -      158

    211514 49765 26971 27450 20353 14118   6759  1526    31    2    1   358490
SDA>
```

2.8 Larger Granularity Hints for Memory-Resident Sections

OpenVMS memory-resident sections on I64 platforms can now use all page sizes that the hardware supports. Using page sizes larger than the OpenVMS default of 8 KB can significantly improve application performance because fewer entries in the translation cache are needed to cover larger amounts of data.

On Alpha and I64 systems, OpenVMS uses the granularity hints feature to implement page sizes larger than the default system page size. On Alpha systems, the maximum effective page size is 4 MB, On current I64 systems, the maximum effective page size is 4 GB. You can use the granularity hints feature for memory-resident sections by preallocating memory using the SYSMAN RESERVED_MEMORY commands. If you want granularity hints, you must use memory-resident sections on Alpha and I64 systems. Use the SYSMAN RESERVED_MEMORY commands to preallocate memory the next time the system boots.

Note the following change in alignment requirements: if you want to use larger page sizes, both virtual and physical addresses must be aligned appropriately. On Alpha systems, sufficient alignment is assured if the section-start address is mapped at a page table boundary. For I64 systems, this might no longer be sufficient.

On OpenVMS I64 systems, HP recommends that you follow these steps to use memory-resident sections:

1. Use the SYSMAN command RESERVED_MEMORY to preallocate memory as needed.
2. Run SYS$UPDATE:AUTOGEN GETDATA REBOOT procedure to tune the system to the reduced amount of memory available for general use.
3. Use shared page table regions for all memory-resident sections. Even if no shared page tables are allocated at boot time, using a shared page table region ensures optimum alignment of the region in virtual address space.

   Do not specify a start address for the shared page table region. This allows OpenVMS to select a start address that permits maximum granularity hint factors for sections mapped within the region.
4. Use the SEC$M_EXPREG flag when mapping the section. This also allows OpenVMS to select the best-aligned start address for the section.
5. After a memory-resident section is created, always map all of the section. If you map only parts of it, you might not be able to use shared page tables or large page sizes, or both.
6. If you cannot change an application that specifies start addresses for shared page table regions or memory resident sections, you might see application failures. As a workaround, you can set bit 4 (a value of 16 decimal) in the dynamic system parameter MMG_CTLFLAGS to select a special operating mode that limits page sizes on I64 systems to the maximum granularity hint size supported on Alpha.

The following sections contain additional changes that are related to the larger granularity hints for memory-resident sections.

2.8.1 Flags Added to MMG_CTLFLAGS System Parameter

The following table shows all the bit mask values currently defined for this parameter, including bit 4, which has been added in this release.

Bit	Meaning
0	If this bit is set, reclamation is enabled by trimming from periodically executing, but otherwise idle, processes. This occurs when the size of the free list plus the modified list drops to less than twice the value of FREEGOAL. This function is disabled if the bit is clear.
1	If this bit is set, reclamation is enabled by outswapping processes that have been idle for longer than LONGWAIT seconds. This occurs when the size of the free list drops below the value of FREEGOAL. This function is disabled if the bit is clear.
2	Controls deferred memory testing on AlphaServer 4100 systems. If this bit is clear (the default), OpenVMS tests memory as a background activity. If the bit is set, all memory is tested during the bootstrap process.
3	Reserved for OpenVMS use; must be zero.
4	If this bit is clear (the default), all page sizes supported by hardware can be used to map resident memory sections on I64. If this bit is set, page sizes on I64 systems are limited to the maximum GH factor available on Alpha systems (512 * <system page size>).
5-7	Reserved for future use.

2.8.2 System Service Changes

The following list contains system services with item code changes that are related to the increase in memory-resident sections:

- $CREATE_REGION_64
  - length_64

    The following information is to appear after the first paragraph of the description:

    If you want to map multiple memory resident sections to this region, specify a length large enough to accommodate not only all of the sections but also to fill the space necessary to align the next section for a maximum of effective page sizes (granularity hints). You can satisfy this requirement by simply allocating a region that is twice as large as the sum of all sections you want to map.
  - start_va_64

    The last paragraph in the description has been changed to the following:

    If the flag VA$M_SHARED_PTS is set and this argument is specified, the specified starting address must be aligned to the larger of a natural page table boundary or the largest possible page size used to map the section. If the alignment is less than a page table boundary, the $CREATE_REGION_64 service returns an error. If the alignment is less than the largest page size used in the section, an error might be returned when you attempt to map the section.

    If you do not specify a starting address, OpenVMS automatically ensures correct alignment.
- $CRMPSC_GDZRO_64
  - section_offset_64

    The last paragraph has been changed to the following:

If a shared page table region is specified by the region_id_64 argument, section_offset_674 must be at least an integer multiple of the larger of the number of bytes that can be mapped by a CPU-specific page table page. For I64 systems, the alignment of section offset must also be an integer multiple of the page size used to map VA space at this offset.

HP recommends that you avoid any partial mapping of memory resident sections when you use shared page tables on I64 systems. If you cannot avoid this, set bit 4 in the system parameter MMG_CTLFLAGS, limiting the effective page size to the number of bytes that can be mapped by a page in a page table.

- start_va_64

  The last paragraph in the description has been changed to the following:

  If the flag VA$M_SHARED_PTS is set and this argument is specified, the specified starting address must be aligned to the larger of a natural page table boundary or the largest possible page size used to map the section. If the alignment is less than a page table boundary, the $CREATE_REGION_64 service returns an error. If the alignment is less than the largest page size used in the section, an error might be returned when you attempt to map the section.

  If you do not specify a starting address, OpenVMS automatically ensures correct alignment.

- $MGBLSC_64
  - section_offset_64

    The last paragraph has been changed to the following:

    If a shared page table region is specified by the region_id_64 argument, section_offset_674 must be at least an integer multiple of the larger of the number of bytes that can be mapped by a CPU-specific page table page. For I64 systems, the alignment of section offset must also be an integer multiple of the page size used to map VA space at this offset.

    HP recommends that you avoid any partial mapping of memory resident sections when you are using shared page tables on I64 systems. If you cannot avoid this, set bit 4 in the system parameter MMG_CTLFLAGS, limiting the effective page size to the number of bytes that can be mapped by a page table page.

  - start_va_64

    The last paragraph in the description has been changed to the following:

    If the flag VA$M_SHARED_PTS is set and this argument is specified, the specified starting address must be aligned to the larger of a natural page table boundary or the largest possible page size used to map the section. If the alignment is less than a page table boundary, the $CREATE_REGION_64 service returns an error. If the alignment is less than the largest page size used in the section, an error might be returned when you attempt to map the section.

    If you do not specify a starting address, OpenVMS automatically ensures correct alignment.

2.9 Partition Manager

Partition Manager, or parmgr, centralizes all nPartition management functions in one place. It provides the system manager with the tools to dynamically reconfigure, power on, power off, create, delete, and modify nPartitions as needed to ensure a smooth and well-controlled operation.

Partition Manager provides system administrators with a convenient graphical user interface to configure and manage nPartitions on HP server systems. Using Partition Manager, you can perform complex configuration tasks without having to remember commands and parameters. You select nPartitions, cells, I/O chassis, or other components from the graphical display, then select an action from a menu.

The Partition Manager runs on both HP-UX and Microsoft® Windows® systems. You can use either version of the Partition Manager to manage nPartitions for OpenVMS I64 Version 8.2–1.

You can use Partition Manager to do the following:

- Create, modify, and delete nPartitions.
- Examine the nPartition configuration of a complex.
- Check the complex for potential configuration and hardware problems.
- Manage hardware resources on the complex.

Partition Manager is a free product that can be downloaded from the following Web site:

`http://www.docs.hp.com/en/PARMGR2/download.html`

Refer to the *HP System Partitions Guide, Administration for nPartitions* for more information about the Partition Manager. You can download the online help from the main Partition Manager Web site or from the following Web site:

`http://www.docs.hp.com/en/PARMGR2/features2.html`

Note this URL is case sensitive, so you must type the address exactly as shown.

2.10 New Power-Saving Feature

On I64 systems, a CPU can be placed in “low-power mode” when it is idle. This minimizes power consumption, thereby reducing energy costs for the system. OpenVMS I64 Version 8.2–1 supports this feature based on the settings of the system parameters CPU_POWER_MGMT and CPU_POWER_THRSH. Descriptions of these two parameters follow:

CPU_POWER_MGMT (dynamic)

A value of 1 for this dynamic parameter means on (the default); a value of 0 means off.

When the CPU_POWER_THRSH parameter value is exceeded, the operating system places an I64 processor in low-power mode if it is idle. OpenVMS I64 does this only if CPU_POWER_MGMT is on. A CPU returns to normal power when it receives an interrupt.

CPU_POWER_THRSH (dynamic)

CPU_POWER_THRSH is a dynamic parameter expressed as a percentage.

OpenVMS I64 monitors how active each CPU is over a fixed time period. If CPU_POWER_MGMT is on and a CPU is idle for a time period indicated by CPU_POWER_THRSH, the CPU is placed in low-power mode if it is idle. A CPU returns to normal power when it receives an interrupt.

For systems supporting real-time operations that require quick response time, HP recommends that this feature be turned off. Use of this feature can result in a small performance degradation.

For more information, refer to the *Intel IA-64 Architecture Software Developer's Manual, Volume 2: IA-64 System Architecture* on the following Web site:

`http:/www.intel.com/design/itanium/manuals/iiasdmanual.htm`

2.11 PPL Units Assignment Tool

The License Management Facility provides a tool, the units assignment tool, to help manage license units for OpenVMS Per Processor Licenses (PPL). The units assignment tool uses comma-separated values (CSV) text files to describe the licensing requirements and assignments across the cluster.

The use of a CSV file allows the file to be opened and updated with any text editor or spreadsheet program and to be fed back into the tool. To activate the units assignment tool, type the following command:

```
$@SYS$EXAMPLES:LMF$PPL_UNITS_ASSIGNMENT.COM
```

For more information, see the PPL online help.

2.12 System Dump Analyzer (SDA) Utility

The following sections list new SDA features provided with the OpenVMS I64 Version 8.2–1 operating system. Refer to Section 2.7 for a description of the SDA command that provides Fibre Channel performance information.

2.12.1 SDA COPY Command Additional Information

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 and, on OpenVMS I64, the unwind data for images activated in processes.

If the dump is being analyzed on the system where it was 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 to 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.

By default, a copy of the original dump, as written at the time of the system crash, will include collection. Use COPY/NOCOLLECT to override this. Conversely, a copy of a dump previously copied by SDA without collection (COPY /NOCOLLECT) does not include collection. Use COPY/COLLECT to override this.

Copying a dump that already contains an appended collection always includes that collection.

For successful collection of all file or unwind data, all disks that were mounted at the time of the system crash should be remounted and be made accessible to the process running SDA. If SDA is invoked early on during startup to save the contents of the dump, for example using CLUE$SITE_PROC (refer to the *HP OpenVMS System Analysis Tools Manual*, Section 2.2.3), but disks are not mounted until a batch job is run, then use the COPY/NOCOLLECT command in the CLUE$SITE_PROC command procedure. Once all disks are mounted, you can use a COPY/COLLECT command to save the file or unwind data.

2.12.2 New Qualifiers for COPY Command

The new qualifiers for the SDA COPY command are as follows:

- /[NO]COLLECT

  The /COLLECT qualifier causes SDA to collect file identification data and unwind data (on OpenVMS I64 systems) from the current system and append it to the dump copy being written. The /NOCOLLECT qualifier prevents SDA from collecting the data and appending it. See the command description for defaults and other details.

- /LOG

  Displays information on the progress of the COPY command - for example, the name of the process being copied in a selective dump or, in the case of COPY/COLLECT on I64, the name of an image whose unwind data is being appended to the dump copy.

2.12.3 New SDA Command

This section describes the new SDA COLLECT command.

COLLECT

Collects file identification to file-name translation data on both Alpha and I64 and process unwind data on I64 only.

Format

COLLECT [/qualifier...]

Parameter

None

Qualifiers

/LOG
Displays information on the progress of the COLLECT command, for example, the name of the process being scanned or, on I64, the name of an image whose unwind data is being collected.

/UNDO
Removes all the file and/or unwind data from an earlier COLLECT command from SDA memory. (COLLECT/UNDO does not affect the file or unwind data already appended to the dump file being analyzed.)

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 and, on OpenVMS I64, the unwind data for images activated in processes.

If the dump is being analyzed on the system where it was originally written, you can use the COLLECT command to collect this data for use in the current SDA session. If the dump is being copied for analysis elsewhere, the COPY/COLLECT command may 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 be made accessible to the process running SDA.

Example

```
SDA>  COLLECT
%SDA-W-DISKNOACC, no access to _$30$DKB100: for file and/or unwind data
%SDA-W-FILENOACC, no access to _$30$DKB0: (7709,1,0) for unwind data
-SYSTEM-W-NOSUCHFILE, no such file
```

In this example, the disk 30DKB100, which was mounted at the time the system crashed, is not available when file or unwind data is being collected. In addition, unwind data cannot be collected for the image with file identification (7709,1,0) on _30DKB0: because it no longer exists.

2.12.4 Changes to SHOW CALL_FRAME Parameter

The **starting address** parameter of the SHOW CALL_FRAME command has the following new description:

For Alpha, an expression representing the starting address of the procedure call frame to be displayed/ For a process that uses pthreads, the following SDA command can be used to display the starting addresses for all pthreads:

```
pthread thread -o u
```

The default starting-address is the contents of the frame pointer (FP) register of the SDA current process.

For I64, the address is one of the following:

- The invocation context handle of a frame
- The address of an exception frame. This is equivalent to:

  ```
  SHOW CALL_FRAME /EXCEPTION_FRAME=address
  ```

- The address of a Thread Environment Block (TEB). For a list of all TEBs for the process, use the following SDA command:

  ```
  pthread thread -o u
  ```

The default starting address is the invocation context handle of the current procedure in the SDA current process.

2.13 Traceback Facility

The Traceback facility for I64 systems includes a new symbolize routine, TBK$I64_SYMBOLIZE. This routine, which can be invoked at any time, allows an application to correlate an executable pc value with the source code which generated that code.

The following information describes this routine.

TBK$I64_SYMBOLIZE

The Traceback symbolize routine, TBK$I64_SYMBOLIZE, allows an application program to identify, or symbolize, the source code that was used to generate a particular executable process space location in the currently running application.

Calling Convention

#pragma pointer_size save #pragma pointer_size 64

```
int32 tbk$i64_symbolize(
   uint64 const pc,
   struct dsc64$descriptor * const filename_desc,
   struct dsc64$descriptor * const library_module_desc,
   uint64 * const record_number,
   struct dsc64$descriptor * const image_desc,
   struct dsc64$descriptor * const module_desc,
   struct dsc64$descriptor * const routine_desc,
   uint64 * const listing_lineno,
   uint64 * const rel_pc);
#pragma pointer_size restore
```

Input

pc	Executable instruction in process to be "tracebacked" by value.

Output

filename_desc	String descriptor in which to return the name of the file containing the code, by reference.
library_module_desc	String descriptor in which to return the name of the text library file containing the code, by reference. Returned only if applicable.
record_number	A 64-bit unsigned integer in which to return the record number (that is, record number n specifies the nth line of the file specified by filename_desc),by reference.
image_desc	String descriptor in which to return the image name, by reference.
module_desc	String descriptor in which to return the module name, by reference.
routine_desc	String descriptor in which to return the routine name, by reference.
listing_lineno	A 64-bit unsigned integer in which to write the compiler listing line number, by reference.
rel_pc	A 64-bit unsigned integer in which to write the relative PC value, by reference.

Return Value

TBK$_NORMAL is returned on successful completion Other unsuccessful completion codes may be returned if an error occurs.

Description

The Traceback facility for I64 systems includes a new symbolize routine, TBK$I64_SYMBOLIZE. This routine, which can be invoked at any time, allows an application to process a condition invoked by an exception in an alternate way. Normal traceback processing generates "traceback stack" information onto SYS$OUTPUT (that is, a series of pc values, one for each stack level). If an application wants to independently process traceback information, it can invoke the traceback handling directly.

An application can specify any executable pc value within its image address space and be returned any or all of the return arguments described in the Output section, above. The return information specifies the location of listing line and/or record number of the source code line that generated the object code that includes the pc value specified. This information can also specify the image name, the module name, and the routine name. A relative pc value, which specifies either the image relative or module relative value, is also returned.

The ability for Traceback to return the values requested is dependent upon whether traceback information was requested during the compilation and linker steps of image generation.

Notes

Specifying any output argument (value) as zero will cause the argument to be ignored.

To link an application that refers to TBK$I64_SYMBOLIZE, include the following within a linker option file:

```
SYS$SHARE:TRACE.EXE/shareable
```

3
OpenVMS V8.2–1 Release Notes

This chapter contains the OpenVMS I64 Version 8.2–1 release notes. Because most of the OpenVMS Version 8.2 release notes still apply, read the release notes that apply to OpenVMS I64 (that is, all notes not marked Alpha only) in the *HP OpenVMS Version 8.2 Release Notes* manual as well as the notes in this chapter before installing or upgrading your system.

The following OpenVMS I64 Version 8.2 release notes do not apply to Version 8.2–1:

- Section 1.9.8, DECnet-Plus Requires a New Version
- Section 2.12, DECnet-Plus for OpenVMS X.25 Data Links Not Supported
- Section 4.10, EFI Tools: VMS_SHOW DUMP_DEV Errors
- Section 5.28.1, General Conditions and Workarounds (I64 Only)

  OpenVMS Debugger problems corrected:

  – Printing array index values in default radix
  – Display of rotating registers

3.1 Adapter Release Notes

The following sections provide release notes for adapters supported with OpenVMS Version 8.2–1.

3.1.1 A9782A, A9784A, and AB465A Restriction in Zoned SAN Environments

In Fibre Channel environments, zoning can be used to place a partition between different environments in your SAN. If any of the combo adapters A9782A, A9784A, and AB465A are mapped in a zone with other Fibre Channel adapters, the system might fail to boot.

3.1.2 Fibre Channel EFI Driver and Firmware Requirements

OpenVMS Version 8.2–1 for Integrity servers requires that the HP A6826A Fibre Channel host-based adapter and its variants have the following minimum version: EFI driver: 1.46 RISC firmware: 3.03.154. For the firmware update, go to the following Web site:

`http://www.hp.com/support/itaniumservers`

Refer to Appendix C of the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*, for more information about configuring and booting Fibre Channel devices

3.2 Associated Product Support

The Software Public Rollout Reports for OpenVMS list the availability of HP software products shipping on the Layered Products Library kits (DVD consolidation) for OpenVMS I64. The reports contain the product name and version, the operating system version required to support the product, and the volume ship date for the product. The information in these tables is continually evolving and is subject to change. The reports are intended for public distribution and are updated monthly. The information is not provided in these release notes because of the changing nature of the information.

The Software Public Rollout Reports for OpenVMS are available from the following web site:

`http://h71000.www7.hp.com/openvms/os/swroll/`

If you do not have Internet access, you can find the operating system support information on any of the quarterly Software Products Libraries in the following files:

```
[README]SW_COMPAT_MATRIX.PS
[README]SW_COMPAT_MATRIX.TXT
```

The Software Public Rollout Reports are also available from your HP support representative.

3.3 Cluster Compatibility Patch Kits

Before introducing an OpenVMS Version 8.2–1 system into an existing OpenVMS Cluster system, you must apply certain patch kits (also known as remedial kits) to your systems running earlier versions of OpenVMS. Note that these kits are version specific.

The versions listed in Table 3–1 are supported in a warranted configuration. For more information about these configurations, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*.

Table 3–1 lists the facilities that require patch kits and the patch kit file names. Each patch kit has a corresponding readme file by the same name with a .README file extension.

You can either download the patch kits from the following Web site, or contact your HP Support representative to receive the patch kits on media appropriate for your system:

`http://www2.itrc.hp.com/service/patch/mainPage.do`

Note

Patch kits are periodically updated on an as-needed basis. Always use the most recent patch kit for the facility, as indicated by the version number in the kit's readme file. The most recent version of each kit is the version posted on the Web site.

Table 3–1 Patch Kits Required for Cluster Compatibility

Facility	Patch Kit File Name
OpenVMS Alpha Version 7.3-2	
Update kit with most patch kits except those listed in this section	VMS732_UPDATE-V0400
C RTL	VMS732_ACRTL-V0100
DCL	VMS732_DCL-V0300
Drivers	VMS732_DRIVER-V0200
Fibre Channel & SCSI	VMS732_FIBRE_SCSI-V0600
Graphics	VMS732_GRAPHICS-V0300
I/O device configuration	VMS732_IOGEN-V0100
Job Controller	VMS732_JOBCTL-V0100
LAN	VMS732_LAN-V0300
Math RTL	VMS732_AMATHRTL-V0100
PCSI	VMS732_PCSI-V0100
SYSMAN and others	VMS732_MANAGE-V0300
Shadowing	VMS732_SHADOWING-V0400[1]
System	VMS732_SYS-V0700
VERIFY	VMS732_VERIFY_V0100
OpenVMS Alpha Version 8.2	
CPU270F functionality	VMS82A_CPU270F-V0100
DECnet-Plus for OpenVMS Alpha ECO1	DNVOSIECO01_V82[1]
Drivers	VMS82A_DRIVER-V0100
Graphics	VMS82A_GRAPHICS-V0100
I/O device configuration	VMS82A_IOGEN-V0100
OpenVMS I64 Version 8.2	
Adapter support (U320 SCSI and Gb Ethernet)	VMS821_AB290A-V0100
DDTM	VMS82I_DDTM-V0100
Drivers	VMS821_DRIVER-V0100
Graphics	VMS82I_GRAPHICS-V0100
I/O device configuration	VMS82I_IOGEN-V0100
License Management Facility	VMS82I_LMF-V0100
MANAGE	VMS821_PERFORMANCE-V0100
MX2CPU functionality	VMS821_MX2-V0100
Runtime library general purpose routines	VMS82I_LIBOTS-V0100

[1]This kit is required if you are running this software in your configuration.

(continued on next page)

Table 3–1 (Cont.) Patch Kits Required for Cluster Compatibility

Facility	Patch Kit File Name
OpenVMS I64 Version 8.2	
Secure Server	VMS82I_SECSRV-V0100
VMS loader	VMS821_IVMSLOA-V0100

3.4 HP OpenVMS Debugger Heap Analyzer Conditions and Workarounds

Condition: The Heap Analyzer startup and the Heap Analyzer START command on the I64 kept debugger (START HEAP_ANALYZER) hangs for threaded applications with upcalls enabled.

Workaround: For threaded or AST-involved applications, HP strongly recommends that you either start the Heap Analyzer *before* setting up any debug events or *after* disabling or canceling those events. (You can enable/reset events after the Heap Analyzer startup and START returns you to debugger control.)

3.5 Documentation Corrections

The following sections describe corrections to various manuals in the OpenVMS documentation set.

3.5.1 *HP OpenVMS Debugger Manual*: Client/Server Interface Correction

The Version 8.2 *HP OpenVMS Debugger Manual* contains incorrect information for the PC client interface kit. This release note corrects the information in Sections 11.1 and 11.2, as follows:

The debug server runs on OpenVMS systems. The client is the user interface to the debugger, from which you enter debugger commands that are sent to the server. The server executes the commands and sends the results to the client for display. The client runs on Microsoft® Windows® 95, Windows 98, Windows NT®, Windows 2000, and Windows XP®.

The correct client kit for the above client platforms is:

```
[DEBUG_CLIENTS011.KIT] DEBUGX86011.EXE
```

There is no special installation procedure for the components that run on OpenVMS. The system administrator should move the OpenVMS debug client kit listed above from the OpenVMS distribution media to a place accessible to PC users, such as a PATHWORKS share or FTP server. The client kit is a self-extracting .EXE file.

Once the appropriate executable file has been transferred to the PC, you can run the file to install the debug client on the PC. The InstallShield installation procedure guides you through the installation.

By default, the debug client is installed in the `\Program Files\OpenVMS` Debugger folder. You can also click Browse to select an alternate location.

The installation creates an OpenVMS Debugger program folder that contains shortcuts to the following items:

- Debug client

- Debug client Help file
- `README` file
- Uninstall procedure

3.5.2 *HP OpenVMS I/O User's Reference Manual*: PTD$READ Clarification

The *HP OpenVMS I/O User's Reference Manual* does not fully describe the behavior of the read request. The following paragraph should replace the first paragraph of Section 6.5.1:

To read data from the pseudoterminal, the control program uses the PTD$READ routine. When a PTD$READ routine is called, the operating system queues a read operation. The read operation completes when the pseudoterminal has characters to output. The read request queries TTDRIVER whether there is data found to be returned. If so, the resulting string of characters is returned. If a read request is issued and no data is available, the read request is queued and then completed at a later time. In this case, the routine always returns at least one character. The read request may complete even when there are no characters available to output. In this rare case when TTDRIVER indicates that there is no more data to be output and there is really no data, the read operation completes with zero bytes of data.

The following paragraphs replace Section D.4.4:

The PTD$READ routine reads data from the pseudoterminal. The read request completes with a minimum of one character and a maximum of the number of characters specified by the **readbuf_len** argument.

When a PTD$READ routine is called, the operating system queues a read operation. The read operation completes when the pseudoterminal has characters to output. The read request queries TTDRIVER whether there is data found to be returned. If so, the resulting string of characters is returned. If a read request is issued and no data is available, the read request is queued and then completed at a later time. In this case, the routine always returns at least one character. The read request may complete even when there are no characters available to output. In this rare case when TTDRIVER indicates that there is no more data to be output and there is really no data, the read operation completes with zero bytes of data.

3.5.3 *HP OpenVMS Version 8.2 New Features and Documentation Overview*: Librarian Utility Corrections

The following release notes provide corrected information about the OpenVMS I64 Librarian utility.

3.5.3.1 /REMOVE Qualifier Correction

In Section 4.8.2.3 of the *HP OpenVMS Version 8.2 New Features and Documentation Overview*, the description of the enhanced library /REMOVE qualifier is incorrect. The correct information follows.

The /REMOVE qualifier has been enhanced for the I64 Librarian utility. The format now allows you to specify the module instance of the symbol to be removed. The enhanced /REMOVE qualifier requests that the LIBRARY command delete one or more entries from the global symbol table of an object library.

3.5.3.2 Accessing ELF Object Libraries Correction

Section 4.8.3.2 of the *HP OpenVMS Version 8.2 New Features and Documentation Overview* contains incorrect information. The following text replaces information in that section:

Accessing ELF Object Libraries

ELF object modules are inherently random access modules, whereas OpenVMS Alpha objects, text modules, and so on, are sequential. To allow random access, a new library routine was created to map the ELF object modules into process P2 space so that applications can make random access queries. To recover virtual address space from this mapping, another library routine was created to remove this mapping. These new routines (LBR$MAP_MODULE and LBR$UNMAP_MODULE) work only with ELF object libraries. These entry points are 64-bit interfaces because they refer to P2 space.

Because of the random-access nature of ELF object files, the following operations are not allowed on ELF object libraries:

LBR$GET_RECORD
LBR$SET_LOCATE
LBR$SET_MOVE

Because inserting modules into the library is a sequential operation, LBR$PUT_RECORD is allowed on ELF object libraries. Because the ELF object modules are not segmented into records, you need to provide the module's on-disk size when calling LBR$PUT_MODULE or upon the first call to LBR$PUT_RECORD when writing a module into the library.

The C code fragment in the following example illustrates how to use LBR$PUT_RECORD to insert an object module:

```
bufdesc->dsc$a_pointer = &p0_buffer ;
bytes_to_transfer = module_size ;

while ( bytes_to_transfer )  {
   transfer = MIN ( bytes_to_transfer ,
                    ELBR$C_MAXRECSIZ ) ;

   bufdesc->dsc$w_length = transfer ;

   status = lbr$put_record (  library_index ,
                            & bufdesc ,
                            & txtrfa ,
                              module_size ) ;
   if ( (status & 1) == 0 )
      break ;

   bytes_to_transfer      -= transfer ;
   bufdesc->dsc$a_pointer += transfer ;
   } ;

if ( (status & 1) == 1 )
   status = lbr$put_end ( library_index ) ;
```

To avoid making several calls to LBR$PUT_RECORD, a new library routine, LBR$PUT_MODULE, has been created.

3.5.4 *HP OpenVMS RTL Library (LIB$) Manual* Corrections

The following sections provides additions and corrections to Version 8.2 of the *HP OpenVMS RTL Library (LIB$) Manual*.

3.5.4.1 *HP OpenVMS RTL Library (LIB$) Manual*: Clarification of Rounding Rule for LIB$CVT_DX_DX

In the description of the LIB$CVT_DX_DX routine in the *HP OpenVMS RTL Library (LIB$) Manual*, the following paragraph under "Guidelines for Using LIB$CVT_DX_DX" should contain specific information about the rounding rule that is used:

Results are always rounded instead of truncated, except for the case described below. Note that loss of precision or range may be inherent in the destination data type or in the NBDS destination size. No errors are reported if there is a loss of precision or range as a result of destination data type.

This paragraph should be modified as follows:

Results are always rounded instead of truncated, except for when the source and destination are both NBDS and no scaling is requested. That case is described more fully in a later rule. LIB$CVT_DX_DX uses the VAX_ROUNDING rule. Note that loss of precision or range may be inherent in the destination data type or in the NBDS destination size. No errors are reported if there is a loss of precision or range as a result of destination data type. For details about the VAX_ROUNDING rule, refer to the description of CVT$CONVERT_FLOAT.

3.5.5 *HP OpenVMS RTL Library (LIB$) Manual*: Clarification of Platform Restrictions

The *HP OpenVMS RTL Library (LIB$) Manual* incorrectly identifies the following routines as being available on both Alpha and I64. These routines are available only on Alpha:

- LIB$GET_CURR_INVO_CONTEXT
- LIB$GET_INVO_CONTEXT
- LIB$GET_INVO_HANDLE
- LIB$GET_PREV_INVO_CONTEXT
- LIB$GET_PREV_INVO_HANDLE
- LIB$PUT_INVO_REGISTERS

The *HP OpenVMS RTL Library (LIB$) Manual* also should specify that the LIB$GET_UIB_INFO routine is available only on I64.

The routines relating to invocation contexts and invocation handles that are I64 only include the following:

- LIB$I64_CREATE_INVO_CONTEXT
- LIB$I64_FREE_INVO_CONTEXT
- LIB$I64_GET_CURR_INVO_CONTEXT
- LIB$I64_GET_CURR_INVO_HANDLE
- LIB$I64_GET_INVO_CONTEXT
- LIB$I64_GET_INVO_HANDLE
- LIB$I64_GET_PREV_INVO_CONTEXT
- LIB$I64_GET_PREV_INVO_HANDLE

For additional information about these routines, refer to the *HP OpenVMS Calling Standard*.

3.5.6 *HP OpenVMS System Manager's Manual*: IPC Commands Restriction

Section 9.15, Using Interrupt Priority Level C (IPC), in the *HP OpenVMS System Manager's Manual, Volume 1: Essentials* incorrectly states that you can use IPC commands on I64 and all Alpha systems. This is not correct. The documentation has been changed to include the following statement:

```
For OpenVMS Versions 8.2 and 8.2--1, you cannot use IPC commands
on I64 systems or on ES47 or GS1280 Alpha systems if you booted
from a Graphic console.
```

3.5.7 *HP OpenVMS System Services Reference Manual* Additions and Corrections

The following sections provide additions and corrections to Version 8.2 of the *HP OpenVMS System Services Reference Manual*.

3.5.7.1 $GETRMI Item Code—RMI$_MODES

Version 8.2 of the *HP OpenVMS System Services Reference Manual* omitted the following information about the new $GETRMI item code, RMI$MODES.

RMI$_MODES allows you to monitor the modes of each CPU in a multi-CPU system (not just the CPU usage of all the nodes together).

RMI$_MODES returns the amount of time, in 10-ms units, spent by all currently active CPUs in all processor modes since the system was booted. Each increment in the returned time for a mode represents an additional 10 ms spent by the CPU in that mode. An active CPU is one that is actively participating in the processor scheduling that the OpenVMS instance performs.

The buffer length field in the item descriptor is dependent on the number of CPUs attached to the system. The size of the buffer passed should be as follows:

```
n * sizeof ( CPU_struct ) + 4
```

where *n* is the available CPU count. Use $GETSYI item code SYI$_ POTENTIALCPU_CNT to get *n*, the count of potentially available CPUs. (Refer to the description of the $GETSYI service for details.)

Data for an individual CPU is returned by means of a CPU_struct structure. Declare the CPU_struct with no member alignment, as in the following example:

```
#pragma member_alignment save
#pragma _nomember_alignment
typedef struct _CPU_struct
{
    unsigned char   cpu_id;     // physical cpu id
    unsigned int    interrupt;  // is actually the sum of interrupt and idle[1]
    unsigned int    mpsynch;    // multi-processor synchronization
    unsigned int    kernel;     // kernel mode
    unsigned int    exec;       // executive mode
    unsigned int    super;      // supervisor mode
    unsigned int    user;       // user mode
    unsigned int    reserved;   // reserved, will be zero
    unsigned int    idle;       // CPU idle
}CPU_struct;
#pragma member_alignment restore
```

---

[1] For compatibility with existing applications, the returned value for interrupt time is the sum of interrupt and idle times. Calculate the actual interrupt time by subtracting the returned value of idle time from the returned value of interrupt time.

For a multiple-CPU system, the data for all active CPUs is returned as an array of type CPU_struct, with the number of elements in the array equal to the number of potentially available CPUs. If a CPU is inactive, the CPU_struct array element corresponding to that CPU will contain 0 in all CPU_struct members.

Be sure to allocate a buffer large enough to hold the counters for all available CPUs.

The following code example shows one method for collecting CPU modes:

```
#include <starlet.h>
#include <syidef.h>
#include <rmidef.h>
#include <efndef.h>
#include <iledef.h>
#include <iosbdef.h>

CPU_struct   *cpu_counters = NULL;
IOSB          myiosb = { 0 };
char         *buffer;
unsigned long buffer_size;
int           status;

// Set up the $GETSYI item list: potential CPUs and active CPU bitmask
// unsigned long
CPU_Count = 0;
unsigned long long ActiveCPUs = 0;

ILE3 SYIitmlst[] = { {sizeof CPU_Count, SYI$_POTENTIALCPU_CNT, &CPU_Count, 0},
                     {sizeof ActiveCPUs, SYI$_ACTIVE_CPU_MASK, &ActiveCPUs, 0},
                     {0,0}};

// Get the available CPU count and a bitmask of active CPUs
status = SYS$GETSYIW( EFN$C_ENF, NULL, NULL, SYIitmlst, &myiosb, NULL, 0 );
buffer_size = CPU_Count * sizeof( CPU_struct ) + 4;
buffer = malloc( buffer_size );

// Set up the $GETRMI item list: CPU modes
ILE3 RMIitmlst[] = {{buffer_size, RMI$_MODES, buffer, 0},
                    {0,0}};

// Call the service
status = SYS$GETRMI( EFN$C_ENF, 0, 0, RMIitmlst, &myiosb, NULL, 0 );

// THE DATA COUNTERS BEGIN 4 BYTES OFF THE START OF THE BUFFER;
// The first 4 bytes of the buffer are reserved for internal use.
cpu_counters = (CPU_struct *) ( buffer + 4 );

// Use the counters for all active CPUs
for ( int i = 0; i < CPU_Count; i++)
{
    if ( 1 == ( 1 & (ActiveCPUs >> i) ) )
    {
       cpu_counters[i].interrupt
       ---
       ---
       ---
    }
}

free( buffer );
```

3.5.7.2 $PUTMSG System Service Correction

The $PUTMSG prototype is incorrectly documented in Version 8.2 of the *HP OpenVMS System Services Reference Manual*. The correct prototype is as follows:

C Prototype

```
int sys$putmsg (void *msgvec, int (*actrtn)(__unknown_params),
void *facnam, unsigned __int64 actprm);
```

Note that the return value from **actrtn* is indeed checked to determine whether or not the message is input.

The documentation source file has been corrected, and the correction will appear in the next version of the *HP OpenVMS System Services Reference Manual* and in online help.

3.5.8 *HP Volume Shadowing for OpenVMS*: Corrections to Memory Requirements

Starting with OpenVMS Version 7.3, additional memory is required to run Volume Shadowing for OpenVMS, as described in the Version 7.3–2 *HP Volume Shadowing for OpenVMS* manual in Chapter 1 under Hardware Environment, Memory Requirements.. Note the following corrections:

- In the sentence, "For example, for a shadow set with 200 GB of storage per member, 420 KB of memory is required for its write bitmaps for every node in the cluster.", the total of 420 should be 400.
- Three paragraphs later, in the sentence, "For example, a system with 10 shadow sets mounted, with each shadow set consisting of 50-GB member disks, would require an additional 1,119 KB of memory.", the total of 1,119 should be 1,069, as it appears in the calculation that follows on the same page.

These totals will be corrected in the next revision of *HP Volume Shadowing for OpenVMS*.

3.6 EFI Shell Precautions on Shared or Shadowed System Disks

On each Integrity system disk, there can exist up to two FAT partitions that contain OpenVMS boot loaders, EFI applications and hardware diagnostics. The OpenVMS bootstrap partition and, when present, the diagnostics partition are respectively mapped to the following container files on the OpenVMS system disk:

SYS$LOADABLE_IMAGES:SYS$EFI.SYS
SYS$MAINTENANCE:SYS$DIAGNOSTICS.SYS

The contents of these FAT partitions appear as fs*n*: devices at the console EFI Shell> prompt. These fs*n*: devices can be directly modified by user command input at EFI Shell> prompt, and by EFI console or EFI diagnostic applications. Neither OpenVMS nor any EFI console environments that might share the system disk are notified of partition modifications; OpenVMS and console environments are entirely unaware of these console modifications. Accordingly, you must ensure the proper coordination and proper synchronization of the changes with OpenVMS and with any other EFI consoles that might be in use.

You must take precautions when modifying the console in configurations using either or both of the following:

- OpenVMS host-based volume shadowing for the OpenVMS I64 system disk
- Shared system disks and parallel EFI console access across Integrity environments sharing a common system disk

You must preemptively reduce these OpenVMS system disk environments to a single-member host-based volume shadowset or to a non-shadowed system disk, and you must externally coordinate access to avoid parallel accesses to the Shell> prompt whenever making shell-level modifications to the fs*n*: devices, such as:

- Installing or operating diagnostics within the diagnostics partition
- Allowing diagnostics in the partition (or running from removable media) to modify the boot or the diagnostic partition on such an OpenVMS I64 system disk
- Otherwise directly or indirectly modifying the boot or the diagnostics partition within these environments from the Shell> prompt

If you do not take these precautions, any modifications made within the fs*n* device associated with the boot partition or the device associated with the diagnostic partition can be overwritten and lost immediately or after the next OpenVMS host-based volume shadowing full-merge operation.

For example, when the system disk is shadowed and changes are made by the EFI console shell to the contents of these container files on one of the physical members, the volume shadowing software has no knowledge that a write was done to a physical device. If the system disk is a multiple member shadow set, you must make the same changes to all of the other physical devices that are the current shadow set members. If this is not done, when a full merge operation is next performed on that system disk, the contents of these files might regress. The merge operation might occur many days or weeks after any EFI changes are done.

Furthermore, if a full merge is active on the shadowed system disk, you should not make changes to either file using the console EFI shell.

To suspend a full merge that is in progress or to determine the membership of a shadow set, refer to the HBMM chapter of the *HP OpenVMS Version 8.2 New Features and Documentation Overview*.

These precautions apply only to Integrity system disks that are configured for host-based volume shadowing, or that are configured and shared across multiple OpenVMS I64 systems. Configurations that are using controller-based RAID, that are not using host-based shadowing with the system disk, or that are not shared with other OpenVMS I64 systems, are not affected.

3.7 Librarian: Increased Object Module Name Key Length

In OpenVMS Version 8.2, the Librarian incorrectly restricted the object module name key length to 31 characters for I64 (ELF) object and shareable image libraries. This problem has been corrected. The Librarian now accepts object module name key lengths of up to 1024 characters.

3.8 Linker Utility

Most of the release notes in the *HP OpenVMS Version 8.2 Release Notes* still apply to the Linker utility in OpenVMS I64 Version 8.2–1. The following changes to the HP OpenVMS Version 8.2 release notes are in effect for OpenVMS Version 8.2–1:

- All release notes in Section 5.24 apply to all OpenVMS platforms instead of Alpha only.
- Section 5.24.4, Limit of 25 Elements on Stack, applies to Alpha and VAX only.
- Section 5.25.4, Errors in the Image Synopsis Section of the Map, no longer applies. The error has been corrected.
- In Section 5.25.5, DIFTYPE and RELODIFTYPE Messages, the introduction to the TYPE1.C and TYPE2.C example and the example should read as follows:

  The following example of two modules demonstrates how to fix these conditions:

```
TYPE1.C

 #include <stdio>

 int status ;   // Defines status as data.
 extern int sub();

 main ()
 {
     printf ("Hello World\n");
     sub();
 }

TYPE2.C

 extern int status (int x) ; //Refers to status as a procedure.

 sub ()
 {
 int x;
         x = (int)status;
         return status (x);
 }
```

The Linker utility new features chapter (Chapter 5) contains the following information that should be understood in light of certain conditions, as follows:

- Section 5.2.2, Handling of Initialized Overlaid Program Sections on OpenVMS

  Condition: Initialized overlaid program sections with zeros are incorrectly seen as compatible initializations. In OpenVMS V8.2 and V8.2–1, the I64 linker accepts sections initialized with zeros to be compatible with sections containing non-zero initial values. Under this condition, the order of modules in a link operation can result in different initial values in the image.

  Resolution: This problem will be fixed in a future release of the I64 linker.

- Section 5.4.4, New Linker Qualifiers: /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS

  Condition: Creating a shareable image by simply exporting all global symbols may not work. The /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS qualifiers assist in creating symbol vector options for all global symbols on a per-module basis. However, using these qualifiers may create new problems over creating a symbol vector in the traditional way.

Resolution: Because of the potential for errors, both qualifiers will be withdrawn in a future release of the I64 Linker.

3.9 Multiple nPartitions on Cell-based Systems

If you have multiple nPartitions on your HP Integrity rx7620, HP Integrity rx8620, or HP Integrity Superdome servers, and you are running a multi-operating system environment with OpenVMS on one of the nPartitions, one of the other operating systems might register an error or event on the System Event Log (SEL) while OpenVMS is booting. OpenVMS holds the SEL until it has produced a table of Field Replaceable Units (FRU), which might cause other operating systems to register an error or an event.

3.10 Network Update Restrictions from Version 8.2 to Version 8.2–1

OpenVMS Version 8.2–1 supports network update of the operating system from Version 8.2 to Version 8.2–1. Network update is supported only over the core I/O LAN cards on systems supported by OpenVMS Version 8.2. Refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual* for more information.

In addition, there is also a hardware configuration restriction for network booting. Unlike Alpha consoles, where the speed and duplex setting for the network adapter can be selected at the console, the Integrity server console and network boot drivers perform autonegotiation only. The network switch nearest the Integrity server boot client must be set to autonegotiate for a successful network boot. Failure to set the switch to autonegotiate results in an inability to complete the network boot process.

3.11 NPAG_AGGRESSIVE and NPAG_GENTLE Default Values

As of OpenVMS Version 8.2, the default values of the NPAG_AGGRESSIVE and NPAG_GENTLE system parameters have been set to the value of 100. A value of 100 turns off both gentle and aggressive reclamation of nonpaged pool lookaside lists. In many cases, when pool reclamation moves small packets from the lookaside lists back to the variable list, this results in fragmentation of the variable list. This fragmentation appears as many small packets at the front of the variable list and a few large packets at the end of the list.

When an allocation occurs for a packet that is larger than any of the lookaside lists, the system must find a large enough packet on the variable list. When heavily fragmented, as described above, the entire variable list often needs to be searched to find a large enough packet. Because the variable list is kept in address order, when a large packet is deallocated, the entire list needs to be searched again to deallocate the packet.

Under these conditions, system performance can be severely degraded. For this reason, HP recommend that you turn off pool reclamation but keep both NPAG_GENTLE and NPAG_AGGRESSIVE system parameters set to 100.

3.12 Synchronous Data Links Not Supported

OpenVMS Version 8.2–1 does not support any synchronous data link hardware on Integrity servers.

3.13 HP TCP/IP Services for OpenVMS

HP strongly recommends that you apply the current Engineering Change Order (ECO) for TCP/IP Services for OpenVMS Version 5.5 after upgrading to OpenVMS Version 8.2–1 to ensure that the latest software is installed.

3.14 Traceback API Problem Fixed

In OpenVMS I64 Version 8.2, an error occurred when the *pc_rel* (relative PC value) or *image_desc* (image name string descriptor) arguments were specified as zero. The Traceback facility now correctly ignores these arguments and continues Traceback processing.

4
InfoServer Utility

This chapter provides a description of the new InfoServer utility feature, including a comparison between the InfoServer hardware and InfoServer application and a reference for the InfoServer utility commands.

4.1 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

4.1.1 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.

4.1.2 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
CODE_EXAMPLE>
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.

5
Linker Utility

This chapter provides an overview of the differences as well as considerations you should review before linking programs on OpenVMS I64 systems as opposed to OpenVMS Alpha.

This chapter discusses the following major topics:

- Differences between linking on I64 systems and linking on Alpha and VAX systems (Section 5.2)
- New aspects of linking an image on OpenVMS I64 (Section 5.3)
- New linker qualifiers and options (Section 5.4, Section 5.5)
- New ways to use existing linker qualifiers and options (Section 5.6)
- Expanded linker map file information for I64 (Section 5.7)

For linker release notes, refer to Chapter 3 and the *HP OpenVMS Version 8.2 Release Notes*, which contain release notes from Version 8.2 that may still apply.

5.1 Linker Utility Overview

The purpose of the linker is to create images (that is, files that contain binary code and data). The linker on OpenVMS I64 is different from the linker on OpenVMS Alpha and VAX systems because it accepts OpenVMS I64 object files and produces OpenVMS I64 images. As on OpenVMS Alpha and VAX systems, the primary type of image created is an **executable image**. This image can be activated at the DCL command prompt using the RUN command.

The OpenVMS I64 Linker also creates **shareable images**. A shareable image is a collection of procedures and data that are exported via the SYMBOL_VECTOR option that can be called by executable images or other shareable images. Shareable images are named in an input options file for inclusion in a link operation that creates an executable or shareable image.

When linking modules, the primary type of input file to the OpenVMS I64 Linker is the **object file**. Object files are produced by language processors such as compilers or assemblers. Because the object files produced by these compilers are unique to the Intel Itanium architecture, some aspects of linking modules on OpenVMS I64 systems are different from linking on Alpha and VAX systems. Overall, however, the OpenVMS I64 Linker interface, as well as functional capabilities (for example, symbol resolution, virtual memory allocation, and image initialization), are similar to those on OpenVMS Alpha and OpenVMS VAX systems.

5.2 Differences When Linking on OpenVMS I64 Systems

Although linking on OpenVMS I64 systems is similar to linking on OpenVMS Alpha systems, some differences exist. The following qualifiers and options are ignored by the OpenVMS I64 Linker:

- /REPLACE
- /SECTION_BINDING
- /HEADER
- DZRO_MIN
- ISD_MAX

The following qualifiers and options are not allowed by the OpenVMS I64 Linker:

- /SYSTEM
- A file specification with the /DEBUG= qualifier
- The base address must be null in CLUSTER=*cluster_name,base_address* . . .
- BASE=*address*
- UNIVERSAL=*symbol-name*

The following keyword has a different meaning for the I64 Linker:

PER_PAGE—The per_page keyword in /DEMAND_ZERO=PER_PAGE directs the linker to compress trailing zeros for each segment (that is, demand zero compression of zeros on trailing pages).

The following list contains new qualifiers that are supported by the OpenVMS I64 Linker:

- /BASE_ADDRESS
- /SEGMENT_ATTRIBUTE
- /FP_MODE
- /EXPORT_SYMBOL_VECTOR
- /PUBLISH_GLOBAL_SYMBOLS
- GROUP_SECTIONS and SECTION_DETAILS keywords for the /FULL qualifier

The following sections describe these qualifiers and options.

5.2.1 Specifying Based Clusters Varies by OpenVMS Platform

Specifying a base address in the CLUSTER option is permitted only on Alpha and VAX systems. On Alpha systems, only main images are allowed to contain based clusters. VAX systems are more flexible; even shareable images are allowed to contain based clusters. On I64 systems, specifying a base address in a CLUSTER option is illegal for all image types.

5.2.2 Handling of Initialized Overlaid Program Sections on OpenVMS I64 Systems

Note

Refer to Section 3.8 for a related release note.

On Alpha and VAX systems, initializations can be done to portions of an overlaid program section. Subsequent initializations to the same portions overwrite initializations from previous modules. The last initialization performed on any byte is used as the final one of that byte for the image being linked.

On I64 systems, the ELF object language currently does not implement the feature of the Alpha and VAX object language which allows the initialization of portions of the program sections. When an initialization is made, the entire section is initialized. Subsequent initializations of this section can be performed only if the nonzero portions match in value. This is called a compatible initialization.

Note

A future release of the I64 Linker will only accept initializations if the initialized portion of the sections match. A section initialized with zeros will no longer match any other section with a nonzero initial value. The initialization in three.c, as shown in Example 5–1, will no longer be compatible with the initializations in sample initializations one.c and two.c.

For example, the following condition produces different results on OpenVMS I64 systems than on Alpha and VAX systems:

Two modules, each declaring two longwords in the same program section, are overlaid. The first module initializes the first longword in the program section with nonzero values. Similarly, the second module initializes the second longword in the program section with nonzero values.

On Alpha and VAX systems, the linker can produce an image section with the first and second longwords initialized. The Alpha and VAX object languages give the linker the section size and the Text Information Relocation (TIR) commands to initialize each longword.

Because there are no TIR commands in ELF on I64 systems, the linker neither performs nor knows about initializations within a section. Rather, the linker receives entire sections from the compilers that are already initialized. The linker reads all the applicable module initializations to the section and checks for compatible initializations. Any two initializations are compatible if they are identical in their nonzero values. If they are not compatible, the linker issues the following error message:

```
%ILINK-E-INVOVRINI, incompatible multiple initializations for
 overlaid section
        section:  <section name>
        module:   <module name for first overlaid section>
        file:  <file name for first overlaid section>
        module:  <module name for second overlaid section>
        file:  <file name for second overlaid section>
```

In this message, the linker lists the first module, which contributes a nonzero initialization, and the first module with an incompatible initialization. Note that this is not a full list of all incompatible initializations; it is simply the first one that the linker encounters.

In the Program Section Synopsis of the linker map, each module with a nonzero initialization is flagged as Initializing Contribution. Use this information to identify and resolve all the incompatible initializations.

Example 5–1 shows the additional information in the map file shown in Example 5–2.

Example 5–1 Additional information

```
$ cre one.c
int common_data[]={0,0,47,11};
void main (void) {return;}
Ctrl/Z
$ cc /extern=common one
$ cre two.c
int common_data[]={0,0,47,11};
Ctrl/Z
$ cc /extern=common two
$ cre three.c
int common_data[]={0,0,0,0,0,0,0,0};
Ctrl/Z
$ cc /extern=common three
$ link/map one,two,three
$
```

Example 5–2 shows the program section synopsis of the linker map for Example 5–1. Note that the Align and Attributes fields normally continue after the Length field but were modified to fit on the page.

Example 5–2 Linker Map Showing Program Section Synopsis

```
                                  +---------------------------+
                                  ! Program Section Synopsis !
                                  +---------------------------+

Psect Name      Module/Image     Base     End          Length
----------      ------------     ----     ---          ------
COMMON_DATA                     00010000 0001001F 00000020 (      32.)
                ONE             00010000 0001000F 00000010 (      16.)
                TWO             00010000 0001000F 00000010 (      16.)
                THREE           00010000 0001001F 00000020 (      32.)

Align                Attributes
-----               ------------
OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,  MOD
OCTA  4 Initializing Contribution
OCTA  4 Initializing Contribution
OCTA  4
```

Example 5–3 shows an incompatible initialization and the resulting linker message.

Example 5–3 Incompatible Initialization

```
$ cre four.c
int common_data[]={0,0,47,11,0,0,17,4};
Ctrl/Z
$ cc /extern=common four
$ link one,two,three,four
%ILINK-E-INVOVRINI, incompatible multiple initializations for
 overlaid section
        section: COMMON_DATA
        module: ONE
        file: DISK$USER:[JOE]ONE.OBJ;1
        module: FOUR
        file: DISK$USER:[JOE]FOUR.OBJ;1
```

Example 5–1 and Example 5–3 were compiled with the extern common model. For OpenVMS, the default extern model is the relaxed reference/definition (ref/def) model. In that model, only one explicit initialization is allowed. That is, even identical initializations result in a linker warning message (that is, the multiply defined message). Example 5–4 uses the same source files as the previous examples.

Example 5–4 Initialization Example on OpenVMS

```
$ cc one
$ cc two
$ cc three
$ link one, two, three
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: TWO
        file: DISK$USER:[JOE]TWO.OBJ;2
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: THREE
        file: DISK$USER:[JOE]THREE.OBJ;2
```

Example 5–5 shows an additional incompatible initialization. In this example, the linker shows both the INVOVRINI error and the MULDEF warning messages. For such a module, the INVOVRINI message precedes the MULDEF message.

Example 5–5 Additional Incompatible Initialization

```
$ cc four
$ link one,two,three,four
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: TWO
        file: DISK$USER:[JOE]TWO.OBJ;2
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: THREE
        file: DISK$USER:[JOE]THREE.OBJ;2
%ILINK-E-INVOVRINI, incompatible multiple initializations for
```

(continued on next page)

Example 5–5 (Cont.) Additional Incompatible Initialization

```
 overlaid section
        section: COMMON_DATA
        module: ONE
        file: DISK$USER:[JOE]ONE.OBJ;2
        module: FOUR
        file: DISK$USER:[JOE]FOUR.OBJ;2
%ILINK-W-MULDEF, symbol COMMON_DATA multiply defined
        module: FOUR
        file: DISK$USER:[JOE]FOUR.OBJ;2
```

5.2.3 Behavior Difference When Linking Relaxed Reference/Definition Symbols

The I64 Linker behaves differently when relaxed reference/definition (ref/def) symbols (also known as "ELF common symbols" symbols on I64) are linked selectively against an image that contains a definition for the same symbol. On Alpha, the linker incorrectly takes the definition from the relaxed ref/def symbol in your module. The I64 Linker takes the definition from the shareable image.

For example, a shareable image is linked from the following module (my_int.c):

```
 #include ints
 uint64 my_int ;
 $ cc/extern=relaxed my_int.c
 $ link/map/full/cross/share my_int,sys$input/opt
 symbol_vector=(my_int=data)
```

Next, another C module (x.c) is compiled and linked selectively against `my_int.exe`. Note that the object is compiled with the relaxed extern model. The result is a conditional ref/def generated for `my_int`.

```
 #include ints
 uint64 my_int;
 main()
 {
 my_int = 1;
 return;
 }

 $ cc/extern=relaxed x.c
 $ link/map/full/cross sys$input/opt
 cluster=myclu,,,x.obj
 my_int.exe/share/select
```

The Alpha Linker incorrectly defines `my_int` from the module's conditional definition of `my_int`. The I64 Linker correctly retrieves the definition from `my_int.exe`.

The Alpha Linker is not being changed to preserve those cases that rely upon this behavior. The I64 Linker performs the selection operation correctly.

5.2.4 Flags Set When /TRACEBACK, /DEBUG, and /DSF Are Used

When the /TRACEBACK, /DEBUG, and /DSF linker debug qualifiers are specified, the flags listed in the following table are set for the image activator. These flags are set in the dynamic segment under the tag value DT_VMS_ LNKFLAGS. Note that the meanings of the /TRACEBACK, /DEBUG, and /DSF qualifiers have not changed for I64, but the meanings and the names of the flags have changed.

Flag	Meaning
VMS_LF_IMGSTA	Image execution is to begin by calling SYS$IMGSTA. The image activator includes SYS$IMGSTA as the first address in the (traditional VMS style) transfer vector.
VMS_LF_CALL_DEBUG	SYS$IMGSTA checks this flag to determine whether it calls the debugger.
VMS_LF_TBK_IN_IMG	Traceback records are present in the image file.
VMS_LF_DBG_IN_IMG	Debug information is present in the image file.
VMS_LF_TBK_IN_DSF	Traceback records are present in the DSF file.
VMS_LF_DBG_IN_DSF	Debug information is present in the DSF file.

The flags are set according to the following table:

Qualifier	IMGSTA	CALL_ DEBUG	TBK_IN _IMG	DBG_IN _IMG	TBK_IN _DSF	DBG_IN _DSF
/NoTrace/NoDebug /NoDSF	0	0	0	0	0	0
/Trace/NoDebug /NoDSF	1	0	1	0	0	0
/NoTrace **/Debug** /NoDSF	1	1	1	1	0	0
/Trace /Debug /NoDSF	1	1	1	1	0	0
/NoTrace /NoDebug **/DSF**	0	0	0	0	1	1
/Trace /NoDebug **/DSF**	1	0	1	0	1	1
/NoTrace **/Debug /DSF**	1	1	1	0	1	1
/Trace /Debug /DSF	1	1	1	0	1	1

Notes

- The value of SYS$IMGSTA is no longer included in the image's transfer array; only a flag that indicates it is to be called. The image activator knows the value of SYS$IMGSTA.
- These flags do not appear in a DSF file. DSF files are not activated by the image activator (they have no dynamic segment and, therefore, no DT_VMS_ LNKFLAGS field).
- The linker no longer supports /DEBUG=*filename* on I64 systems. Link alternative debuggers as a separate image and then define LIB$DEBUG to point to that image.
- When /DSF is specified along with /TRACEBACK or /DEBUG, the VMS_LF_ TBK_IN_IMG (traceback in image) flag is set. This is a change in behavior from Alpha, where traceback records are not included in the image when /TRACEBACK/DSF or /DEBUG/DSF is specified. Note that debugger records do not get copied to an image whenever /DEBUG/DSF is specified. Here, /DEBUG causes only the IMGSTA bit to be set in the image.

The following table indicates where global symbol definitions are written during a link operation that uses the debugging qualifiers:

Qualifier	Global Symbols in Image	Global Symbols in DSF File
/NoTrace/NoDebug /NoDSF	0	0
/Trace /NoDebug /NoDSF	0	0
/NoTrace **/Debug** /NoDSF	1	0
/Trace /Debug /NoDSF	1	0
/NoTrace /NoDebug **/DSF**	0	1
/Trace /NoDebug **/DSF**	0	1
/NoTrace **/Debug** **/DSF**	0	1
/Trace /Debug **/DSF**	0	1

5.3 New Aspects for Linking on OpenVMS I64 Systems

The following sections describe aspects of linking images on I64 systems that are unique to the platform. Topics include:

- Understanding linkage messages (Section 5.3.1)
- Considerations for images compiled with reduced floating-point model (Section 5.3.2)
- Considerations for linking with ELF groups and UNIX-style weak symbols (Section 5.3.3)
- Use of the new /BASE_ADDRESS qualifier (Section 5.4.1)
- Use of the new /SEGMENT_ATTRIBUTE qualifier (Section 5.4.2)
- Use of the new /FP_MODE qualifier (Section 5.4.3)
- Use of the new /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS qualifiers (Section 5.4.4)
- Use of the new GROUP_SECTIONS and SECTION_DETAILS keywords for the /FULL qualifier (Section 5.4.5)
- New alignments for the PSECT_ATTRIBUTE option (Section 5.5.1)

5.3.1 Understanding Linkage Messages

To determine consistent use of the general registers across a call, some HP compilers produce call linkage information that allows the linker to verify the linkage of a caller to the linkage of the called routine. Linkage information is supplied for only Intel Itanium general registers 1 through 31.

The following example shows a warning message that displays when the linker detects a linkage conflict:

```
%ILINK-W-LNKGERR, linkage to routine X is not compatible with
 linkage of caller
        calling module: MOD_SRC
                file: DISK:[DIR]SOURCE.OBJ;1
        target  module: MOD_TARG
                file: DISK:[DIR]TARGET.OBJ;1
        source type of JSB to target type of CALL
        register AI not provided (needed at target)
        register GP not provided (needed at target)
        IA64 register R19 (Alpha R21) -- call=PRESERVE, target=VOLATILE
```

Following the display of the source and target information are messages that indicate the nature of the conflict or conflicts that caused the warning message to be displayed. If a caller does not provide argument information in the argument information register (AI) when the target routine requires the AI, the linker generates a message similar to the one in this example.

In addition to missing required information, inconsistent or incompatible use of the general registers can also cause a linkage conflict. The linkage information contains the following register policies for the general registers:

- Volatile: A register with this policy can not be used to pass information between procedures, either as input or output.
- Scratch: A register with this policy can be modified by the called procedure.
- Output: A register with this policy can be used to pass information back to the calling procedure.
- Preserve: A register with this policy must have its contents saved and then restored, if it is to be used by the target routine.

The register policies of a standard linkage call are as follows:

Intel Itanium Register	Policy
R2	Volatile
R3	Scratch
R4—R7	Preserve
R8—R9	Output
R10—R11	Scratch
R12—R18	Volatile
R19—R24	Scratch
R26—R31	Scratch

A standard procedure call uses the CALL mechanism and provides the Global Pointer (GP) value and the AI register filled with the argument information.

The following table indicates compatible policies between the caller's register policies (row heading) and the target routine's register policies (column heading):

Caller	Target			
	Volatile	Scratch	Output	Preserve
Volatile	X			
Scratch		X		X
Output			X	X
Preserve				X

In the following sample message, Intel Itanium register R19 has a caller register policy of Preserve but a target routine register policy of Volatile.

```
IA64 register R19 (Alpha R21) - call=PRESERVE, target=VOLATILE
```

Based on the preceding table, R19 is not compatible between the caller and the target routines.

Additionally, the calling mechanism between the routines is not compatible. This occurs because the caller is performing a JumptoSuBroutine (JSB) to the target routine when the target routine is expecting a CALL mechanism to be used.

Certain Intel Itanium registers must be defined to have specific policies. If these registers do not contain the correct policies, the linker issues a warning indicating that either the target linkage is invalid (1) or the caller linkage is invalid (2):

```
(1) %ILINK-W-INVLNKG, invalid target linkage for symbol INV_LNKG
(2) %ILINK-W-INVRLNKG, invalid call linkage for symbol INV_LNKG
```

The following table lists the Intel Itanium general registers that have specific policies:

Intel Itanium Register	Policy
R2	Volatile
R12—R18	Volatile

Explanation of Potential Conflicts that Result in Linkage Messages

Some conflicts arise when performing cross-language calls (for example, IMACRO to BLISS), where the calling standards for OpenVMS differ between Alpha and I64 systems. There are fewer registers preserved by default with the Intel Itanium version of the calling standard. Because of this, there may be assumptions about registers that are saved at the target routine and registers that are not saved.

Linkage statement mismatches also cause the linker to display warning messages. For example, if a calling routine is expecting register R12–R15 (Alpha register numbers) to be preserved but the target does not preserve them, the linker lists registers R20, R21, R30, and R31 as having a linkage problem but includes the Alpha register numbers in parentheses.

Additionally, a problem can occur when registers that are not properly declared for the function they are serving, for example, declaring registers to return values as NOPRESERVE rather than OUTPUT.

To avert these potential conflicts, examine the declaration and definition linkages for all cases that the linker indicates, and correct these problems to ensure a clean link operation. Because the linker considers these conflicts as warnings, the completion code status of the image is not set to SUCCESS. For a shareable

image, this means that images linked against a shareable image with linkage issues receive a completion status other than SUCCESS.

5.3.2 Considerations for Images Compiled with Reduced Floating–Point Model

On OpenVMS I64 systems, images using a reduced floating-point model run more quickly during interrupts because only a reduced number of floating-point registers need to be saved and restored. Note that some integer arithmetic is done using floating-point registers. The resulting image can be run in a reduced floating-point mode only if all object modules that contribute to the image have been compiled with the reduced floating-point model. The Object and Image Synopsis section of the linker map file indicates whether the modules have the reduced floating-point model.

5.3.3 Considerations for Linking with ELF Groups and UNIX-Style Weak Symbols

The Intel C++ compiler introduced the use of Executable and Linking Format (ELF) Groups (also known as COMDATs) and UNIX-style weak symbols that the linker (and the Librarian) must correctly handle.

Background

ELF groups in object modules can be seen as tentative code and data contributions with definitions for procedures and variables. The C++ compiler makes use of this feature for C++ templates. The compiler simply generates code and data as a group. In such an environment, multiple object modules have multiple identical contributions with the same code and data. For the image, the linker selects one contribution for each group. The result of linking multiple object modules with multiple identical groups is one instance of code and data per group in the image.

An ELF group has a name, also called its group signature. Groups are considered identical if they have the same name. Symbols can belong to a group; they are called group symbols.

Understanding UNIX-style Weak Symbols, Definitions, and References

Group symbols can be used as definitions or references. However, UNIX-style weak definitions differ from the VMS-style weak definitions in that there can be multiple UNIX-style weak definitions. UNIX-style weak references are similar to Alpha and VAX weak references in that when they are left unresolved, they do not result in an error or warning message. In ELF, the weak references and definitions of the Alpha and VAX object language are referred to as VMS-style weak symbols.

The linker takes the first occurrence of the specific group, along with that group's normal and UNIX-style weak symbols, as the defining instance of that group unless it is overridden by a shareable image symbol. All subsequent definitions of these UNIX-style weak symbols from other occurrences of that group are then seen as references to the selected instance. UNIX-style weak symbols are tentative by design. If present and there is no occurrence of a strong definition elsewhere among the other modules or images of the link, the first encountered UNIX-style weak version of the symbol is regarded as the defining instance.

A strong definition overwrites a UNIX-style weak definition, which becomes a reference. The other UNIX-style weak symbols of the same group become references. This does not create a problem because all symbols, including strong definitions, are generated by the compiler or defined in the run-time environment. By understanding this mechanism, you can better understand the linker map.

UNIX-style weak symbols defined in a group can be referenced by nonweak symbols outside a group.

Reading the Linker Map

The cross-reference list in the map has a UNIX-style weak tag in front of UNIX-style weak definitions and references. Typically, the UNIX-style weak tagged definitions are the result of selecting a group from an object module. The UNIX-style weak tagged references are usually a result of a secondary occurrence of the same group. An untagged definition with a UNIX-style weak tagged reference is usually a result of a strong definition overwriting a group symbol from an object module. The tagging in the cross reference table enables the user to see which module was selected as owning the defining instance of the symbol.

The cross-reference does not list the group signatures (that is, their names). The Analyze utility can be used to find all symbols that belong to a group. Additionally, you can ask the linker for a list of all groups and their defining modules and files by using the /MAP/FULL=GROUP_SECTIONS qualifiers.

Considerations for Shareable Image Linking

User-written templates can be linked as a shareable image. Symbols that are exported (universalized) by a shareable image are always strong definitions. With the precedence of strong definitions over UNIX-style weak ones, the code and data of the shareable image is the linker selected contribution. For OpenVMS I64, the grouping of symbols is preserved in the shareable image as well. In this case, where all the symbols are strong definitions, a group from a shareable image always takes precedence over all identical groups in object modules.

There is one difference between groups in a shareable image and groups in an object module. If the same group is seen in another shareable image, the linker issues an error message and does not write an image file.

A user error results if code and data from the first occurrence of the group is used in all object modules within the defining shareable image, but code and data from the other occurrences is used within the other defining shareable images. Therefore the linker does not produce an image.

When linking a shareable image, make sure that all symbols of all groups making up the template implementation are exported. Otherwise, redundant code and data might appear in the resulting image or, as mentioned previously, wrong code or data might be used.

The original OpenVMS-style weak symbols continue to be processed in the same manner as in previous releases.

5.4 New Linker Qualifiers for OpenVMS I64

Note

Refer to Section 3.8 for a related release note.

Some new linker qualifiers have been added to support linking on OpenVMS I64 systems. This section describes these qualifiers and associated keywords.

5.4.1 New /BASE_ADDRESS Qualifier

The /BASE_ADDRESS qualifier assigns a virtual address for images that are not activated by the OpenVMS image activator, such as images used in the boot process. The base address is the starting address that you want the linker to assign to an executable image. The OpenVMS image activator is free to ignore any linker-assigned starting address. This qualifier is used primarily by system developers.

The /BASE_ADDRESS qualifier does not replace the base-address specifier in the CLUSTER=[*base-address*] option, which is illegal on OpenVMS I64. (Refer to Section 5.2.1.)

5.4.2 New /SEGMENT_ATTRIBUTE Qualifier

The /SEGMENT_ATTRIBUTE qualifier sets certain attributes for segments. It uses the following syntax:

```
/SEGMENT_ATTRIBUTE=(segment_attribute [, ...])
```

The OpenVMS I64 Linker accepts DYNAMIC=*address_region*, SHORT=WRITE, CODE=*address_region*, and SYMBOL_VECTOR=[NO]SHORT as segment attributes, where an address region can be specified with keywords P0 and P2.

By default, the linker puts the dynamic segment, which contains information for the image activator, into P2 space. For images not activated by the OpenVMS image activator, DYNAMIC=P0 forces the linker to put the dynamic segment into P0 space. This qualifier is primarily used by system developers.

The SHORT_DATA=WRITE keyword allows you to combine the read-only and the read-write short data segments into a single segment, reclaiming up to 65,535 bytes of unused, read-only space (default value for /BPAGE). When setting SHORT_DATA to WRITE, your program may accidentally write to formerly read-only data. Therefore, this qualifier is recommended only for users whose short data segment has reached the limit of 4 MB.

With the CODE=P2 keyword, the I64 Linker allows you to assign code segments to P2 space. When the image activator activates the image, the code segments will be placed in P2 space. If you use this keyword, be aware that all code addresses will be 64 bits wide. For example, your exception handlers should use only the 64-bit versions of the signal and mechanism arrays and should be prepared to handle a 64-bit PC.

By default, for shareable images, the linker stores the symbol vector into the read-only short data segment. By specifying SYMBOL_VECTOR=NOSHORT, the linker collects the symbol vector into a read-only data segment of the default cluster. If the shareable image has none, such a segment is created. This frees up the short data of the symbol vector entries.

5.4.3 New /FP_MODE Qualifier

The OpenVMS I64 Linker determines the program's initial floating-point mode using the floating-point mode provided by the module that provides the main transfer address. The /FP_MODE qualifier sets an initial floating-point mode only if the module that provides the main transfer address does not provide an initial floating-point mode. The /FP_MODE qualifier does not override an initial floating-point mode provided by the main transfer module.

The OpenVMS I64 Linker accepts the following keywords to set the floating-point mode:

- D_FLOAT, G_FLOAT—Sets VAX floating-point modes.
- IEEE_FLOAT[=*ieee_behavior*]—Sets the IEEE floating-point mode to the default or a specific behavior.

The OpenVMS I64 Linker accepts the following IEEE behavior keywords:

- FAST
- UNDERFLOW_TO_ZERO
- DENORM_RESULTS (default)
- INEXACT

The OpenVMS I64 Linker also accepts a floating-point mode behavior literal. For more information about the initial floating-point mode, Refer to the *HP OpenVMS Calling Standard*.

5.4.4 New Linker Qualifiers: /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS

The /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS qualifiers were added to the linker to aid users who are creating shareable images but do not know which symbols to export through the SYMBOL_VECTOR option. For example, if you are porting an application from UNIX and are unfamiliar with the application, you might not know which symbols to export. Another scenario is that you are coding in C++ and you are not able to correlate the mangled name with the source code symbol you want to export.

Note

Be aware that exporting all global symbols may have unintended consequences.

Exporting all symbols is a normal practice for creating shared objects on UNIX systems. On OpenVMS systems, however, the mechanisms of linking and image activation are different. Therefore, employing a UNIX mechanism to create an OpenVMS shareable image may not work. Conflicts may arise when linking an application against multiple shareable images that were generated to export all global symbols. These shareable images may contain user- or compiler-generated symbols of the same name. When the linker tries to resolve a module's reference to one or more of these symbols, the link operation will fail if it finds definitions for a symbol in more than one shareable image. Even when you restrict symbol exporting to just a few select modules (or even one module), you may still encounter these conflicts.

Although the /EXPORT_SYMBOL_VECTOR and /PUBLISH_GLOBAL_SYMBOLS qualifiers were created to assist you port open source software, these conflicts may create new problems over creating a symbol vector in the traditional way. For these reasons, both qualifiers will be withdrawn in a future release of the I64 Linker.

The new /PUBLISH_GLOBAL_SYMBOLS qualifier marks an object file or library object module so that all its global symbols are exported in the file with symbol vector options generated by the linker. The new /EXPORT_SYMBOL_VECTOR qualifier writes to the output file a symbol vector option for each global symbol in modules marked with /PUBLISH_GLOBAL_SYMBOLS. When the /EXPORT_SYMBOL_VECTOR qualifier is present, only the options file is written; no image file is generated. You must complete the generated options file with the GSMATCH information before you use it in a future link operation.

Both qualifiers are accepted only if the /SHAREABLE qualifier is present. /EXPORT_SYMBOL_VECTOR is a command line only qualifier. The /PUBLISH_GLOBAL_SYMBOLS qualifier can be used from the command line and in options files. The linker issues a warning if you specify an /EXPORT_SYMBOL_VECTOR qualifier but do not specify a /PUBLISH_GLOBAL_SYMBOLS qualifier on the command line or in options files.

/EXPORT_SYMBOL_VECTOR=[file-spec]

The /EXPORT_SYMBOL_VECTOR qualifier instructs the I64 Linker to create an options file with symbol vector options that are filled in as directed by the /PUBLISH_GLOBAL_SYMBOLS qualifier, plus a template GSMATCH option.

Using the /EXPORT_SYMBOL_VECTOR qualifier inhibits image production.

If the /EXPORT_SYMBOL_VECTOR qualifier is present, input SYMBOL_VECTOR= option clauses are not allowed.

At least one /PUBLISH_GLOBAL_SYMBOLS qualifier must be on the command line or in an option.

The qualifier accepts a file specification that you want the linker to use as the name of the generated options file. If you do not specify a file type, the linker assumes a type of .OPT.

If you append the /EXPORT_SYMBOL_VECTOR qualifier to an input file specification, the linker creates an options file using the file name of the file to which the qualifier is appended.

After filling in the GSMATCH template option, use the generated options file in another link operation to create the shareable image file.

/PUBLISH_GLOBAL_SYMBOLS

The /PUBLISH_GLOBAL_SYMBOLS qualifier instructs the I64 linker to mark an object file, a library object module, or an object module library (that is, all modules extracted from the library) to have their global symbols exported via symbol vector clauses in the linker-generated options file.

The qualifier can be used only with the /SHAREABLE and /EXPORT_SYMBOL_VECTOR qualifiers.

The qualifier is compatible with the /INCLUDE and /SELECTIVE_SEARCH qualifiers. The qualifier can be used on the command line and in a linker options file.

If the qualifier is applied to an object file, all global symbols from all modules in that file are candidates to be exported as a symbol vector option.

If the qualifier is applied to an object file in combination with the /SELECTIVE_SEARCH qualifier, only referenced global symbols from all modules in that file are candidates to be exported as a symbol vector option.

If the qualifier is applied to an object library, all global symbols from all modules of the object library, which would be implicitly included in the image, are candidates to be exported as a symbol vector option.

If the qualifier is applied to an object library in combination with the /INCLUDE qualifier, all global symbols from the modules in the include list of the object library are candidates to be exported as a symbol vector option.

If the qualifier is applied to an object library built with the /SELECTIVE_ SEARCH qualifier, only referenced global symbols from all modules of the object library, which would be included in the image, are candidates to be exported as a symbol vector option. (Refer to the *HP OpenVMS Command Definition, Librarian, and Message Utilities Manual* for how to insert modules in a library that can be searched selectively.)

If a global symbol is a tentative definition in several modules (most likely from a relaxed ref/def extern model), then the symbol is published if at least one of the tentatively defining modules is tagged with the /PUBLISH_GLOBAL_SYMBOLS qualifier. The same is true for UNIX-style weak symbol definitions in several modules. In both cases, however, encountering a module with a strong symbol definition overrides the tentative definition. The symbol's export disposition is determined by the presence or absence of the /PUBLISH_GLOBAL_SYMBOLS qualifier on the overriding module.

Examples

```
$ link/SHARE public/PUBLISH,implementation/EXPORT=public
```

In this example, all the global symbols from the modules in PUBLIC.OBJ are exported as a symbol vector option to the file PUBLIC.OPT.

```
$ LINK/SHAREABLE api_table,implementation/PUBLISH/SELECTIVE/EXPORT=public
```

In this example, only the global symbols from all the modules in IMPLEMENTATION.OBJ, which have been previously referenced are exported as a symbol vector option to the file PUBLIC.OPT.

```
$ LINK/SHAREABLE api_table,plib/LIBRARY/PUBLISH/EXPORT
```

In this example, all the global symbols from all the modules of the library PLIB.OLB, which would be included in the shareable image, are exported as a symbol vector option to the file PLIB.OPT.

```
$ LINK/SHAREABLE plib/PUBLISH/INCLUDE=public/EXPORT
```

In this example, all the global symbols from the module PUBLIC of the library PLIB.OLB are exported as a symbol vector option to the file PLIB.OPT.

```
$ LINK/SHAREABLE api_table,
public/PUBLISH/SELECTIVE/EXPORT=public
```

In this example, all the global symbols from all the modules in PUBLIC.OBJ that have been referenced after PUBLIC.OBJ has been processed are exported as a symbol vector option to the file PUBLIC.OPT. A similar example with /PUBLISH being used in an options file, follows:

```
$ link/SHAREABLE TT:/opt/EXPORT
public/PUBLISH, implementation
Ctrl/Z
```

In this example, all the global symbols from the modules in PUBLIC.OBJ are exported as a symbol vector option to TT: that is printed to the terminal.

5.4.5 New GROUP_SECTIONS and SECTION_DETAILS keywords for the /FULL Qualifier

The OpenVMS I64 Linker takes two keywords to the /FULL qualifier that directs the linker to create a full image map.

/FULL [=(*keyword* [,...])]

The OpenVMS I64 Linker accepts the following keywords to tailor the map (the default is /FULL=SECTION_DETAILS):

Keyword	Meaning
GROUP_SECTIONS	Directs the OpenVMS I64 Linker to list all processed groups (ELF COMDATs).
SECTION_DETAILS	Directs the OpenVMS I64 Linker to list zero-length contributions in the Program Section Synopsis.
ALL	The ALL keyword is equivalent to specifying both the GROUP_SECTIONS and SECTION_ DETAILS keywords.

The GROUP_SECTIONS keyword directs the linker to include, in the map file, all groups used in the link operation. Currently, only the C++ compiler takes advantage of groups. Using this keyword with other languages has no effect.

When /FULL=NOSECTION_DETAILS is specified, the OpenVMS I64 Linker suppresses zero length contributions in the Program Section Synopsis of the map. The /FULL qualifier defaults to /FULL=SECTION_DETAILS. A full linker map on I64, Alpha, and VAX systems lists all the module contributions in the Program Section Synopsis.

5.5 Extensions to Linker Options for OpenVMS I64

The following extensions to existing linker options have been added to support linking on OpenVMS I64 systems.

5.5.1 New Alignments for the PSECT_ATTRIBUTE Option

The PSECT_ATTRIBUTE option now accepts integers 5, 6, 7, and 8 for the alignment attribute. The integers represent the byte alignment indicated as a power of 2. (For example, 6 represents a 2 ** 6 (64-byte) alignment.) The keyword HEXA (for hexadecimal word) was added for 2 ** 5 (that is, a 32-byte or 16-word alignment).

5.6 New Ways to Use Existing Linker Qualifiers and Options

The following sections describe new ways to use existing linker qualifiers and options on I64 systems. Topics include:

- Using mixed-case arguments in linker options (Section 5.6.1)
- Conventions for specifying image names (Section 5.6.2)
- Using the PSECT_ATTRIBUTE option to specify alignment (Section 5.6.3)

5.6.1 Mixed-Case Arguments in Linker Options on I64 Systems

On OpenVMS I64 systems, names issued by compilers can be mixed-case names. If you need to operate on mixed-case names in the options file, the linker provides the CASE_SENSITIVE option that allows you to specify names in mixed case, rather than using the default (all names in uppercase). For example, if you have a library include statement and the module names in the library are mixed-case, you can set the CASE_SENSITIVE option as follows to allow mixed-case names:

```
CASE_SENSITIVE=YES
```

When the CASE_SENSITIVE option is set to YES, all characters to the right of the left-most equals sign in the option clause (that is, the option arguments) have their case preserved. These characters are taken as is, without modification: file names, module names, symbol names, and keywords. To restore the linker's default behavior of uppercasing the entire option line, specify the CASE_ SENSITIVE option with the NO keyword. For example:

```
CASE_SENSITIVE=NO
```

Note that the NO keyword must appear in uppercase or the linker will not recognize it.

For example, the following options file contains mixed-case names that you want to preserve by setting the linker to case-sensitive mode:

```
case=Yes
My_Lib/library/include=(Add_Func, Sub_Func)
symbol_vector=(Add_Func=PROCEDURE,PAGE_COUNT=DATA)
case=NO
```

When processed by the linker, the text appears as follows:

```
CASE=YES
MY_LIB/LIBRARY/INCLUDE=(Add_Func,Sub_Func)
SYMBOL_VECTOR=(Add_Func=PROCEDURE,PAGE_COUNT=DATA)
CASE=NO
```

The case of all names to the right of the first equal sign in each option remains the same.

To maintain Alpha and VAX behavior, HP recommends that you switch to case sensitivity only when needed.

5.6.2 Conventions for Specifying Image Names

The following conventions describe the various names that apply to an image:

- Images are given an image file specification (for example, FOO.EXE) that can be changed with the DCL command RENAME.
- The image name is specified with the NAME= option and stored in the image in a note of type NT_VMS_IMGNAM. This name can be different than the image file specification name. However, if you do not use the NAME= option, the name defaults to the image file specification name. The Analyze utility displays this name as the "Image name". You cannot change this name with the DCL command RENAME.
- An additional name for the image is associated with the global symbol table (GST) and stored in the image in a note of type NT_VMS_GSTNAM. The linker sets this name to be the same as the image file specification name. This name is used by the Librarian when you insert an image into an image

library. It is displayed by the Analyze utility as the Global Symbol Table Name. You cannot change this name with the DCL command RENAME.

On Alpha systems, the image name specified with NAME= is used to identify self-references in the shareable image list. Self-references are calls from within the image to itself, by means of an alias name in the symbol vector.

On I64 systems, there is no entry in the shareable image list for the current image. Self-references are referred to with a special index value into the shareable image list (-1 in the DT_VMS_FIXUP_NEEDED field) that results in a set of DT_NEEDED entries.

5.6.3 Using the PSECT_ATTRIBUTE Option to Specify Alignment

Do not specify a smaller section alignment with the PSECT_ATTRIBUTE option than the alignment that the compiler gave to the section.

If you specify a smaller alignment for section than any compiler-assigned alignment from all contributions to this section, the linker now issues a warning. For example:

```
$ link hi,sys$input/opt
psect_attr=$literal$,byte
%ILINK-W-CONFALGN, PSECT option alignment (1) less than compiler
assigned (16);
alignment ignored
        section: $LITERAL$
        module: HI
        file: DISK$USER:[JOE]HI.OBJ;3
```

For the Alpha and VAX systems, the linker inappropriately aligns the program section on the boundary that you specified ("byte", in the preceding code example), and places all the contributions to that program section (from other modules you might have linked with "HI", in the example) on boundaries that were not specified by the compiler. The linker does not issue an error message.

The linker always aligns sections, at a minimum, on the boundary specified by the compiler.

The PSECT_ATTRIBUTE option aligns the section on the specified boundary when it is equal to or greater than that which the compiler specified. It does not align each individual contribution to the section; rather, it aligns the total section. The PSECT_ATTRIBUTE option follows the compiler's alignment specification when it aligns each individual contribution.

5.6.4 Special Handling of Linker Nonexistent Files

When the RMS_RELATED_CONTEXT linker option is on (the default is RMS_RELATED_CONTEXT=YES) and a nonexistent file is specified in a list of files for the LINK command, the linker's call to LIB$FIND_FILE may take a long time to complete and the linker may appear to hang. Depending on the number of files being linked and the use of logical names in their specification, the linker may take hours to finish because LIB$FIND_FILE locates every combination of the missing file's prefix before displaying a "file not found" message. Note that you cannot terminate the linker process by pressing Ctrl/Y after the linker has called LIB$FIND_FILE.

To determine which file is missing, follow these steps:

1. Specify SYS$INPUT:/OPTION in the LINK command and press Return. (The linker waits for you to enter option clauses for the link operation from the terminal.)

2. Enter the option clauses and include the following information:
   - On the first line, specify this line:

     ```
     RMS_RELATED_CONTEXT=NO
     ```

     With the RMS_RELATED_CONTEXT option set to NO, any missing file listed in this options file generates an immediate "file not found" message.
   - On subsequent lines, specify the files to be linked, using full file specifications (in the form *disk*:[*dir*]*filename.ext*) for every file. Full file specifications are required because when you specify RMS_RELATED_CONTEXT=NO, file name "stickiness" is disabled.
3. Press Ctrl/Z.

For example, consider the following LINK command:

```
$ LINK DSK:[TEST]A.OBJ, B.OBJ
```

If you want to specify this command with RMS_RELATED_CONTEXT=NO, specify SYS$INPUT:/OPTION and then enter full file specifications for the files to be linked:

```
$ LINK SYS$INPUT:/OPTION
RMS_RELATED_CONTEXT=NO
DSK:[TEST]A.OBJ, DSK:[TEST]B.OBJ
Ctrl/Z
```

Example

The following example shows how the linker appears to hang when file DOES_NOT_EXIST.OBJ is included in the list and the RMS_RELATED_CONTEXT option is not specified (and, therefore, defaults to YES).

```
$  DEFINE DSKD$ WORK4:[TEST.LINKER.OBJ.]
$  DEFINE RESD$ ROOT$, ROOT2$, ROOT3$, ROOT4$, ROOT5$, DISK_READ$:[SYS.] ❶
$  DEFINE ROOT$ WORK4:[TEST.PUBLIC.TEST]
$  DEFINE ROOT2$ WORK4:[TEST.LINKER.]
$  DEFINE ROOT3$ WORK4:[TEST.UTIL32.]
$  DEFINE ROOT4$ WORK4:[TEST.PUBLIC.]
$  DEFINE ROOT5$ WORK4:[TEST.PUBLIC.TMP]
$  LINK/MAP/FULL/CROSS/EXE=ALPHA.EXE  RESD$:[TMPOBJ] A.OBJ,-
_$  RESD$:[SRC]B.OBJ,C,DSKD$:[OBJ]D.OBJ,E,RESD$:[TMPSRC]F.OBJ,-
_$  RESD$:[TEST]G.OBJ,RESD$:[SRC.OBJ]H,RESD$:[COM]DOES_NOT_EXIST.OBJ
Ctrl/T NODE6::_FTA183: 15:49:46 LINK CPU=00:02:30.04 PF=5154 IO=254510 MEM=134 ❷
Ctrl/T NODE6::_FTA183: 15:49:46 LINK CPU=00:02:30.05 PF=5154 IO=254513 MEM=134
Ctrl/T NODE6::_FTA183: 15:50:02 LINK CPU=00:02:38.27 PF=5154 IO=268246 MEM=134
Ctrl/T NODE6::_FTA183: 15:50:02 LINK CPU=00:02:38.28 PF=5154 IO=268253 MEM=134
Ctrl/T NODE6::_FTA183: 15:50:14 LINK CPU=00:02:44.70 PF=5154 IO=278883 MEM=134
```

❶ These commands define logical names and equivalents.

❷ Each time you press Ctrl/T, the CPU and IO values increase, but the MEM and PF values do not, indicating that LIB$FIND_FILE has been called.

As shown in the following example, using an options file to set RMS_RELATED_CONTEXT to NO causes the link operation to finish immediately when it encounters the missing file.

```
$  DEFINE DSKD$ WORK4:[TEST.LINKER.OBJ.]
$  DEFINE RESD$ ROOT$, ROOT2$, ROOT3$, ROOT4$, ROOT5$, DISK_READ$:[SYS.]
$  DEFINE ROOT$ WORK4:[TEST.PUBLIC.TEST.]
$  DEFINE ROOT2$ WORK4:[TEST.LINKER.]
$  DEFINE ROOT3$ WORK4:[TEST.UTIL32.]
$  DEFINE ROOT4$ WORK4:[TEST.PUBLIC.]
$  DEFINE ROOT5$ WORK4:[TEST.PUBLIC.TMP.]
$  LINK/MAP/FULL/ CROSS /EXE=ALPHA.EXE SYS$INPUT:/OPTION
   RMS_RELATED_CONTEXT=NO
   RESD$:[TMPOBJ]A.OBJ,RESD$:[SRC]B.OBJ,RESD$:[SRC]C,DSKD$:[OBJ]D.OBJ
   DSKD$:[OBJ]E,RESD$:[TMPSRC]F.OBJ,RESD$:[TEST]G.OBJ
   RESD$:[SRC.OBJ]H,RESD$:[COM]DOES_NOT_EXIST.OBJ
Ctrl/Z

%LINK-F-OPENIN, error opening DISK_RESD$:[SYS.][COM]DOES_NOT_EXIST.OBJ; as input
-RMS-E-FNF, file not found
$
```

5.7 New OpenVMS I64 Linker Map

The linker map has been enhanced with new information for OpenVMS I64 Linker. The linker map is comprised of the following information, shown in the following sample map:

- Object and image synopsis
- Cluster synopsis
- Image segment synopsis
- Program section synopsis
- Symbol cross reference
- Symbols by value
- Image synopsis
- Link run statistics

Callout descriptions of new and change portions of the linker map are provided after the example linker map.

Figure 5–1 Object and Image Synopsis and Cluster Synopsis

```
                                                            28-OCT-2004 13:34        Linker I02-17

                                          +---------------------------+
                                          ! Object and Image Synopsis ! (1)
                                          +---------------------------+

Module/Image (2)  File            Ident           Attributes (3)     Bytes  Creation Date      Creator
------------      ----            -----        ----------------      -----  -------------      -------
GETJPI                            V1.0             Lkg     Dnrm        360  28-OCT-2004 13:32  HP C V7.1-005
                  SYS$SYSROOT:[SYSMGR]GETJPI.OBJ;1
DECC$SHR                          V8.2-00          Lkg                   0  21-OCT-2004 11:23  Linker T02-17
                  SYS$COMMON:[SYSLIB]DECC$SHR.EXE;1
SYS$PUBLIC_VECTORS                X-3          Sel Lkg                   0  21-OCT-2004 11:23  Linker T02-17
                  SYS$COMMON:[SYSLIB]SYS$PUBLIC_VECTORS.EXE;1


                 Key for Attributes
                +-------------------------------------------+
                ! Sel  - Module was selectively searched    !
                ! Lkg  - Contains call linkage information  !
                ! Dnrm - Denormal IEEE FP model             !
                +-------------------------------------------+


                                               +-------------------+
                                               ! Cluster Synopsis ! (4)
                                               +-------------------+

Cluster (5)                       Match (6)    Majorid   Minorid
-------                           -----        -------  ----------
MYCLU
DEFAULT_CLUSTER
DECC$SHR                          LESS/EQUAL         1           1
SYS$PUBLIC_VECTORS                EQUAL           9114  3906113603
```

VM-1173A-AI

Figure 5–2 Image Segment Synopsis

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34      Linker I02-17

                                  +-------------------------+
                                  ! Image Segment Synopsis ! 7
                                  +-------------------------+

Seg#  Cluster          Type     Pglts   Base Addr   Disk VBN   PFC   Protection   Attributes
----  -------          ----     -----   ---------   --------   ---   ----------   ----------
   0  MYCLU            LOAD         1   00010000           2     0   READ WRITE
   1                   LOAD         1   00020000           0     0   READ WRITE   DEMAND ZERO
   2                   LOAD         1   00030000           3     0   READ ONLY    EXECUTABLE,SHARED
   3                   LOAD         1   00040000           4     0   READ ONLY    SHARED
   4                   LOAD         1   00050000           5     0   READ ONLY    [UNWIND]
8  5  DEFAULT_CLUSTER  LOAD         1   00060000           6     0   READ ONLY    SHORT
   6                   DYNAMIC      2   Q-00000000
                                        80000000           7     0   READ ONLY

                  Key for special characters above
                    +-------------------------+
                    !  Q  - Quadword         !
                    +-------------------------+
```

VM-1174A-AI

Figure 5–3 Program Section Synopsis

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1                28-OCT-2004 13:34     Linker I02-17
                                  +-----------------------------+
                                  ! Program Section Synopsis !
                                  +-----------------------------+

  Psect Name     Module/Image      Base     End      Length      Align      Attributes 9
  ----------     ------------      ----     ---      ------      -----      ----------

  ITMLST                       00010000 0001000F 00000010 (        16.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,  MOD
                 GETJPI        00010000 0001000F 00000010 (        16.) OCTA  4 Initializing Contribution 11

  FILLEN                       00020000 00020003 00000004 (         4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker> 10   00020000 00020003 00000004 (         4.) OCTA  4

  FILLM                        00020010 00020013 00000004 (         4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker>      00020010 00020013 00000004 (         4.) OCTA  4

  IOSB                         00020020 00020027 00000008 (         8.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker>      00020020 00020027 00000008 (         8.) OCTA  4

  STATUS                       00020030 00020033 00000004 (         4.) OCTA  4 OVR,REL,GBL,NOSHR,NOEXE,  WRT,NOVEC,NOMOD
                 <Linker>      00020030 00020033 00000004 (         4.) OCTA  4
```

VM-1175A-AI

Figure 5–4 Program Section Synopsis (Continued)

```
$CODE$                  00030000 0003015F 00000160 (    352.) OCTA  4 CON,REL,LCL,  SHR,  EXE,NOWRT,NOVEC,  MOD
        GETJPI          00030000 000300FF 00000100 (    256.) OCTA  4
        <Linker>  12    00030100 0003015F 00000060 (     96.) OCTA  4

$LINK$                  00040000 00040000 00000000 (      0.) OCTA  4 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,NOMOD
        GETJPI          00040000 00040000 00000000 (      0.) OCTA  4

$LITERAL$               00040000 00400017 00000018 (     24.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00040000 00400017 00000018 (     24.) OCTA  4

$READONLY$              00040020 0004002F 00000010 (     16.) OCTA  4 CON,REL,LCL,  SHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00040020 0004002F 00000010 (     16.) OCTA  4

$LINKER  UNWIND$        00050000 00050017 00000018 (     24.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00050000 00050017 00000018 (     24.) QUAD  3

$LINKER  UNWINFO$       00050018 0005002F 00000018 (     24.) QUAD  3 CON,REL,LCL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD
        GETJPI          00050018 0005002F 00000018 (     24.) QUAD  3

$LINKER  SYMBOL_VECTOR$ 00060000 00060007 00000008 (      8.) OCTA  4 CON,REL,GBL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD,SHORT
        <Linker Option> 00060000 00060007 00000008 (      8.) OCTA  4

$LINKER  SDATA$         00060008 000600AF 000000A8 (    168.) OCTA  4 CON,REL,GBL,NOSHR,NOEXE,NOWRT,NOVEC,  MOD,SHORT
        <Linker>        00060008 000600AF 000000A8 (    168.) OCTA  4
```

VM-1176A-AI

Figure 5–5 Symbol Cross Reference

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1                28-OCT-2004 13:34    Linker I02-17


                          +-----------------------------+
                          ! Symbol Cross Reference !
                          +-----------------------------+

Symbol          Value           Defined By          Referenced By ...
------          -----           ----------          ----------------
DECC$TXPRINTF   00000496-X (13) DECC$SHR            GETJPI
ELF$TFRADR      00060050-R      WK-GETJPI
FILLEN          00020000-R      GETJPI              GETJPI
FILLM           00020010-R      GETJPI              GETJPI
GETJPI  (U)     00000000        <Linker Option>
INTERNAL_GETJPI 00060098-R      GETJPI              GETJPI
IOSB            00020020-R      GETJPI              GETJPI
ITMLST          00010000-R      GETJPI
STATUS          00020030-R      GETJPI              GETJPI
SYS$GETJPIW     0000009A-X      SYS$PUBLIC_VECTORS  GETJPI
```

VM-1177A-AI

Figure 5–6 Symbols by Value

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34    Linker I02-17


                                          +--------------------+
                                          ! Symbols By Value !
                                          +--------------------+

        Value       Symbols...
        -----       ----------
        00000000    GETJPI (U)
        0000009A    X-SYS$GETJPIW
        00000496    X-DECC$TXPRINTF
        00010000    R-ITMLST
        00020000    R-FILLEN
        00020010    R-FILLM
        00020020    R-IOSB
        00020030    R-STATUS
        00060050    R-ELF$TFRADR
        00060098    R-INTERNAL_GETJPI

                                          Key for special characters above (14)
                                          +----------------------+
                                          !   *  - Undefined     !
                                          !  (U) - Universal     !
                                          !   R  - Relocatable   !
                                          !   X  - External      !
                                          !   C  - Code Address  !
                                          !  WK  - Weak          !
                                          ! UxWk - Unix-Weak     !
                                          +----------------------+
```

VM-1178A-AI

Figure 5–7 Image Synopsis

```
SYS$SYSROOT:[SYSMGR]GETJPI.EXE;1          28-OCT-2004 13:34      Linker I02-17

                                  +--------------------+
                                  ! Image Synopsis !
                                  +--------------------+

                 Virtual memory allocated:        00010000 0006FFFF 00060000 (393216. bytes, 768. pages)
                 64-Bit Virtual memory allocated: 00000000 00000000 00000000
                                                  80000000 80010000 00010000 (65536. bytes, 128. pages)
                 Stack size:                                  0. pages
                 Image header virtual block limits:           1.       1. (    1. block)
                 Image binary virtual block limits:           2.       8. (    7. blocks)
                 Image name and identification:       GETJPI V1.0
                 Number of files:                             5.
                 Number of modules:                           3.
                 Number of program sections:                  9.
                 Number of global symbols:                 3338.
                 Number of cross references:                 17.
                 Number of image segments:                    7.
                 Transfer address from module:           GETJPI
                 User transfer FD address:        00000000 00060050
                 User transfer code address:      00000000 00030000
                 Initial FP mode:                 00000000 09800000 (IEEE DENORM_RESULTS)
                 Number of code references to shareable images:    3332.
                 Image type:                      SHAREABLE. Global Section Match=EQUAL, Ident, Major=9120, Minor=2068313277
                 Reduced Floating Point model (RFP): Image does not use RFP model
                 Map format:                      FULL WITH CROSS REFERENCE in file SYS$SYSROOT:[SYSMGR]GETJPI.MAP;1
                 Estimated map length:            441. blocks
```

VM-1179A-AI

Figure 5–8 Link Run Statistics

```
+-------------------------+
! Link Run Statistics !
+-------------------------+

Performance Indicators                Page Faults      CPU Time       Elapsed Time
----------------------                -----------      --------       ------------
   Command processing:                         55  00:00:00.00    00:00:00.00
   Pass 1:                                    173  00:00:00.06    00:00:00.05
   Allocation/Relocation:                       5  00:00:00.02    00:00:00.02
   Pass 2:                                     32  00:00:00.01    00:00:00.00
   Symbol table output:                         4  00:00:00.00    00:00:00.00
   Map data after object module synopsis:       5  00:00:00.00    00:00:00.07
Total run values:                             274  00:00:00.09    00:00:00.17

Quota usage (15)                      ByteCount  FileCount  PgFlCount
------------                          ---------  ---------  ---------
   Available:                            127808        100     512000
   Command processing:                      384          2       7888
   Pass 1:                                  384          2      10240
   Allocation/Relocation:                   576          3      10240
   Pass 2:                                  576          3      18624
   Symbol table output:                     384          2      18624
   Map data after object module synopsis:   384          2      18624

Using a working set limited to 16384 pages and 11029 pages of data storage (excluding image)

Number of modules extracted explicitly          = 0
   with 0 extracted to resolve undefined symbols


1 library searches were for symbols not in the library searched

A total of 8 global symbol table entries was written

LINK/DEB/MAP/FULL/CROSS/SHARE GETJPI.OPT/OPT
<SYS$SYSROOT:[SYSMGR]GETJPI.OPT;2>
cluster=myclu,,,getjpi.obj
symbol_vector=(getjpi/internal_getjpi=procedure
```

VM-1180A-AI

The following descriptions correspond to the callout numbers in the preceding linker map figures:

❶ Object and Image Synopsis. The Alpha section titled Object Module Synopsis is renamed on I64 to Object and Image Synopsis.

❷ Module/Image. The Alpha column Module Name is renamed on I64 to Module /Image.

❸ Attributes. The new Attributes column is added to display four subcolumns of existing or new module attributes.

The first of the four subcolumns indicates whether the symbol search of the module was selective. If the symbol search was selective, the abbreviation Sel appears. If the symbol search of the module was not selective, this subcolumn is left blank.

The second subcolumn indicates whether the module has call linkage information. If the module has linkage information, Lkg appears. If the module does not have linkage information, this subcolumn is left blank.

The third subcolumn indicates whether the module was compiled with the Reduced Floating-Point model. If it was, RFP appears. If the module was not compiled with the Reduced Floating-Point model, this subcolumn is left blank. This designation is suppressed for shareable images.

The fourth subcolumn indicates the Floating-Point mode for the module. Several abbreviations can appear in this column. An explanation of the abbreviations used appears in the Key for Attributes legend that appears at the end of the Object and Image Synopsis section. The following example lists all of the possible abbreviations for this subcolumn in the Keys for Attributes legend. The Creation Date and Creator columns are omitted from this example; refer to the preceding map example for the entire Object and Image Synopsis.

```
Module/Image      File           Ident              Attributes       Bytes
------------      ----           -----           ----------------    -----
NONE                              V1.0               Lkg              568
                  DISK1:[JOE]NONE.OBJ;1
NOFLOAT_CASE                                         Lkg RFP          504
                  DISK1:[JOE]NOFLOAT.OBJ;1
DNORM_CASE                                           Lkg     Dnrm     504
                  DISK1:[JOE]DENORM_W.OBJ;1
FAST_CASE                                            Lkg     Fast     504
                  DISK1:[JOE]FAST_W.OBJ;1
NEPCT_CASE                                           Lkg     Inex     504
                  DISK1:[JOE]INEXACT_W.OBJ;1
SPCL_CASE                                            Lkg     Spcl     504
                  DISK1:[JOE]SPECIAL_W.OBJ;1
UNDER_CASE                                           Lkg     Undr     504
                  DISK1:[JOE]UNDERFLOW_W.OBJ;1
DG_FL_CASE                                           Lkg     VXfl     504
                  DISK1:[JOE]VAXFLOAT_W.OBJ;1
DECC$SHR                          V8.2-00            Lkg                0
                  RESD$:[SYSLIB]DECC$SHR.EXE;1
SYS$PUBLIC_VECTORS                X-2              Sel Lkg              0
                  RESD$:[SYSLIB]SYS$PUBLIC_VECTORS.EXE;1
```

```
 Key for Attributes
+-----------------------------------------+
! Sel  - Module was selectively searched   !
! Lkg  - Contains call linkage information !
! RFP  - Conforms to the reduced FP model  !
! VXfl - VAX Float FP model                !
! Dnrm - Denormal IEEE FP model            !
! Fast - Fast IEEE FP model                !
! Inex - Inexact IEEE FP model             !
! Undr - Underflow-to-zero IEEE FP model   !
! Spcl - Special FP model                  !
+-----------------------------------------+
```

❹ Cluster Synopsis. The Alpha section titled Image Section Synopsis is divided into two sections for I64: Cluster Synopsis and Image Segment Synopsis. The Cluster Synopsis section no longer contains image sections, which are referred to as segments on I64. Further, image sections are no longer printed for shareable images included in the link operation, which is done in VAX map files because there is a possibility of having based shareable images. (Based shareable images are not allowed on Alpha or I64 systems.)

❺ Cluster. The Cluster column shows clusters that were created for and used by the linker, and the order in which they were processed.

❻ Match, Majorid, and Minorid. The Match, Majorid, and Minorid columns show the Global Section Match (GSMATCH) criteria along with the major and minor version numbers, if this information is available.

❼ Image Segment Synopsis. The Image Segment Synopsis section shows each image segment as it was created. It contains the remaining columns of the Image Section Synopsis on OpenVMS Alpha. The first column, Seg #, contains the image segment's number, which is used in the relocations that are applied to it. (Refer to an analysis of an image for a display of the segment number in relocations.) The Alpha section Protection and Paging has been divided into two columns for I64 systems: Protection and Attributes. The column Global Section Name is eliminated.

❽ If the module was compiled /TIE and the image is linked /NONATIVE_ONLY and if the image contains nonstandard signatures, a separate segment might appear immediately after the short data segment (indicated by SHORT) that contains them.

❾ The section attributes PIC, NOPIC are not valid attributes on I64 and are removed.

❿ The linker contributes storage for common or relaxed ref/def symbols. It is marked with <Linker> under the Module/Image header. The section name is always named after the symbol. (This module was compiled with the default switch /EXTERN=RELAXED, and the variables ITMLST, FILLEN, FILLIM and IOSB are relaxed ref/def symbols).

⓫ The linker indicates which modules made initializations (if there were any) to sections which have the attributes OVR, REL and GBL with the designation Initializing Contribution. If you get multiple initialization errors, the linker will have two or more sections marked with the designation Initializing Contribution, in order to help you debug an instance that has many contributors.

⓬ The linker makes a contribution to the code segment containing trampolines (instructions with larger branches) or code to branch to another segment (either inside or outside the image). It is marked with <Linker> under the Module/Image header.

⓭ The designation of an external symbol is changed from Alpha maps. The prefix or suffix used on Alpha is RX, meaning relocatable and external. However, the linker does not know whether or not an external symbol is relocatable or not. As a result, on I64 systems the prefix or suffix is changed to X (external).

⓮ Keys for Special Characters. The keys for special characters are changed as follows:

- On I64, the special character C appears for code address. When a function does not have a function descriptor assigned by the linker, its value is its code address.
- On OpenVMS Alpha, a universal symbol appears once with its internal value. The map never displays an external, universal value (its index into the symbol vector on I64). Universal symbols on I64 appear once with a suffix of (U) defined by <Linker Option> to indicate the external value, and again, possibly with the prefix or suffix R, to indicate their internal value. If you had a symbol vector with an alias name, the alias name appears with the universal value, and the internal name appears with the internal value. The prefixes and suffixes A and I (for Alias and Internal) on OpenVMS Alpha are removed for I64.

  For example, `symbol_vector=(getjpi/internal_getjpi=procedure)` yields:

```
00000000        GETJPI (U)
00050098      R-INTERNAL_GETJPI
```

- UNIX-style weak symbols, designated by UxWk, are a new addition to OpenVMS I64. They are similar to OpenVMS weak symbols; however, more than one symbol with a UNIX-style weak definition can be processed when linking multiple modules without producing a multiple definitions error. UNIX-style weak symbols are currently produced by the C++ compiler.

⓯ Quota Usage. A new section titled Quota Usage is added to the Link Run Statistics section to keep track of the quotas that are being used by the I64 Linker. If quota issues occur, the linker is usually able to work around them. But the linker outputs a special message to the Quota Usage section indicating what quota should be increased to improve performance. For example:

```
Performance of this link operation could be improved by increasing quotas
   Quota related to status return:  %SYSTEM-SECTBLFUL, process or global
   section table is full
2688 extra file I/O operations performed due to current process quota(s)
36 performed on object files; 2652 performed on library files
```

6
Product Directories

This chapter contains information about the OpenVMS Version 8.2–1 media kit, including:

- Pointers to installation information
- Directory structure of the OpenVMS I64 Operating Environment DVD
- OpenVMS Freeware CDs and pointers to related information
- OpenVMS Open Source Tools CD, containing OpenVMS ports of various Open Source Projects not included elsewhere in the OpenVMS kit
- Product licensing
- Documentation

6.1 OpenVMS I64 Operating Environment

This section describes where the OpenVMS I64 operating environment is located on the DVD and lists the names and locations of all products that ship on the OpenVMS I64 OE DVD.

To install the OpenVMS I64 operating environment, refer to the *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*. In addition, before you install the OpenVMS I64 operating environment, review the release notes in this manual and in the *HP OpenVMS Version 8.2 Release Notes* manual.

6.1.1 Directories on the OpenVMS I64 Operating Environment DVD

The top-level directories, documentation directories, and OpenVMS documentation file names for OpenVMS Software Product Descriptions (SPDs) are listed in Table 6–1.

Table 6–1 Directory Structure on the OpenVMS I64 Operating Environment DVD

Directory	File Name/Contents
[.000000]	I64 8.2–1 OS
[.AVAILMAN_I640241]	Availablity Manager Version 2.4–1
[.VMSI18N_I640821]	C/C++ I18N
[.CDSA_I64021]	CDSA Version 2.1
[.CSWS_JAVA_I64021]	CSWS_JAVA (aka Tomcat) Version 2.1
[.CSWS_PERL_I640111]	CSWS_PERL Version 1.1–1
[.CSWS_PHP_I640121]	CSWS_PHP Version 1.2.1
[.DCE_I64032]	DCE Version 3.2
[.DCPS_I64024A]	DCPS Version 2.4A
[.DECNET_PHASE_IV_I640821]	DECnet Phase IV Version 8.2–1
[.DECNET_PLUS_I64_V0821] *includes FTAM, OSAK and VT*	DECnet-Plus Version 8.2–1
[.DWMOTIF_I64015]	DECwindows Motif Version 1.5
[.ENTERPRISE_DIR_I64054]	Enterprise Directory Version 5.4ECO1
[.JAVA_I640142]	Java™ Version 1.4.2–1
[.KERBEROS_I64021]	Kerberos Version 2.1
[.MGMTAGENTS_I64032]	Management Agents Version 3.2
[.NETBEANS_I64036]	NetBeans Version 3.6
[.PERL_I640561]	Perl Version 5.6.1
[.SWB_I64014]	Secure Web Browser Version 1.4
[.SWS_I640131]	Secure Web Server Version 1.3–1
[.SOAP_020]	SOAP Toolkit Version 2.0
[.SSL_I64012]	SSL Version 1.2
[.TCPIP_I64055]	TCP/IP Services for OpenVMS Version 5.5
[.TDC_I64021]	TDC (The Performance Data Collector) Version 2.1-XX
[.UDDI_I64010]	UDDI4J Version 1.0
[.XML_I64020]	XML–J Version 2.0
[.XML_I64020]	XML–C Version 2.0
[.I640821.DOCUMENTATION	Product documentation in .PS and .TXT formats: OVMS_V821_INSTALL.[PS,TXT] OpenVMS Upgrade and Installation Manual OVMS_V821_NEW_FEATURES_REL_ NOTES.[PS,TXT] OpenVMS New Features and Release Notes OVMS_V821_SPD.[PS,TXT] OpenVMS SPD CLUSTER_SPD.[PS,TXT] OpenVMS Cluster Software SPD DECram_SPD.[PS,TXT] DECram Software SPD VOLUME_SHADOWING_SPD.[PS,TXT] Volume Shadowing for OpenVMS SPD

6.2 Directories on the OpenVMS Freeware CD

Three OpenVMS Freeware Version 7.0 CDs are included in the OpenVMS Version 8.2–1 media kit. The Freeware CDs contain a variety of unsupported software tools and utilities, and OpenVMS ports of common open-source packages. These packages can be useful for a variety of technical, engineering, or educational purposes.

For information on each specific freeware package, refer to the appropriate FREEWARE_README.TXT file. To access the file, insert the appropriate CD into the CD drive and enter the following commands, as appropriate to the Freeware volume being mounted. Within the MOUNT commands shown, the ddcu: specification is the device name of the particular CD or DVD device present on your OpenVMS system.

```
$ MOUNT/OVERRIDE ddcu:
$ TYPE ddcu: [FREEWARE_README.TXT
```

Once the appropriate CD is mounted, you can view the disk contents and directory structure using standard DCL commands (such as the DIRECTORY command), or you can use one of the following commands to invoke the Freeware menu system for the appropriate volume:

```
$ @DISK$FREEWARE70_1:[FREEWARE]FREEWARE_MENU

$ @DISK$FREEWARE70_2:[FREEWARE]FREEWARE_MENU

$ @DISK$FREEWARE70_3:[FREEWARE]FREEWARE_MENU
```

For package-specific licensing and related information, consult the individual packages.

Copies of these packages and packages on older Freeware distributions, submission information for future Freeware distributions, and any updated information for this Freeware distribution are all available at:

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

6.3 Open Source Tools CD

The Open Source Tools CD is a collection of Open Source tools ported to OpenVMS by OpenVMS engineering. These open source tools are provided as free software under the terms of the GNU Lesser General Public License. You can redistribute them and modify them under the terms of the GNU Lesser General Public License as published by the Free Software Foundation Version 2.1 of the License.

HP distributes this library in the hope that it will be useful; however, HP provides NO WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For more details, refer to the GNU Lesser General Public License, which is located in the GNV kit directory on the Open Source Tools CD.

The Open Source Tools CD contains the following ports:

- GNV—an open source, GNU-based, UNIX environment for OpenVMS that provides UNIX application developers, system managers, and users a UNIX-style environment. This facilitates development and porting of UNIX software to OpenVMS. (GNU is a UNIX-like operating system that is free software.) GNV provides a UNIX-like shell (command-line interpreter) environment and a C Run-Time Library (CRTL) supplemental library to provide utilities

typically found on UNIX systems. The shell used by GNV is bash (Bourne-Again SHell, from GNU, using the POSIX.2 specification). The Open Source Tools CD contains two GNV kits: one for OpenVMS Alpha and one for OpenVMS I64.

- IAS (Intel Itanium Assembler/Deassembler)—An OpenVMS I64 port of the open source Itanium assembler available from Intel. It can be used to write low-level Itanium assembler code. A few additional features were added to the assembler to make it more usable on OpenVMS I64. Additional information about the new features and how to use the assembler is in the kit.
- STUNNEL—A program that allows you to encrypt arbitrary TCP connections inside an SSL (Secure Sockets Layer) connection from your OpenVMS system to another machine. Stunnel enables you to secure non-SSL aware applications (such as Telnet, IMAP, and LDAP) by providing the encryption and not requiring changes to the original application. Both images and source are provided.

The Open Source Tools CD also contains:

- SSL (Secure Sockets Layer) Sources—included as an optional layered product with OpenVMS Version 8.2–1. The CD provides the SSL Version 1.2 for OpenVMS port sources.
- CD-Record Sources—included as a part of OpenVMS Version 8.2–1. The full source kit used to create the CD record images is provided in the operating system.
- GnuPG (GNU Privacy Guard)—GNU's tool for secure communication and data storage. It can be used to encrypt data and to create digital signatures. GnuPG includes an advanced key management facility. GnuPG is a complete and free replacement for PGP. Because it does not use the patented IDEA algorithm, it can be used without any restrictions. GnuPG is an RFC 2440 (OpenPGP) compliant application.
- CDSA (Common Data Security Architecture) Sources—Included as part of OpenVMS Version 8.2–1. The CD provides the full source kit used to create CDSA for OpenVMS Version 2.1.
- Kerberos Sources—Included as part of OpenVMS Version 8.2–1. The CD provides the full kit used to create Kerberos for OpenVMS Version 2.1.
- GTK+—A open source, free software library for creating graphical user interfaces.
- libIDL—The IDL compiling library, which is an open source, free software library used for creating tress of CORBA Interface Definition Language (IDL) files.
- Freeware tools tar and zip in the [.000TOOLS] directory.

6.3.1 Directories on the Open Source Tools CD

Table 6–2 lists the contents and directories on the OpenVMS Open Source Tools Version 3.0 CD.

Table 6–2 OpenVMS Open Source Tools Version 3.0 CD

Product	Directory
CD-Record Sources	[.CDRECORD_SOURCE]
CDSA Sources	[.CDSA_SOURCE]
Decompression (unzip) and other tools	[.000TOOLS]
GnuPG	[.GNUPG]
GNV for I64 Version 1.6–4	[.GNV_I64]
GTK+	[.GTK]
IAS (Intel Assembler Source)	[.IAS]
Kerberos Sources	[.KERBEROS_SOURCE]
libIDL	[.LIBIDL]
SSL Sources	[.SSL_SOURCE]
Stunnel	[.STUNNEL]

6.4 Product Licensing

The software contained on the OpenVMS I64 media belongs to HP. Use of the software is authorized only if you have a valid software license from HP for each of the products.

The License Management Facility (LMF) Product Authorization Key (PAK) allows you to access a software product. A PAK must be registered and loaded before you can install the associated software from this CD. To obtain PAKs, contact your HP support representative or your authorized reseller.

Licensing for OpenVMS I64 differs from licensing for OpenVMS Alpha in that the OpenVMS operating system and certain layered products are provided in operating environments: the Foundation Operating Environment (FOE); the Enterprise Operating Environment (EOE); and the Mission Critical Operating Environment (MCOE), which will be available soon. The EOE contains the full contents of the FOE plus additional layered products. The MCOE will contain the full contents of the EOE plus OpenVMS Clusters and Reliable Transaction Router (RTR).

You can purchase the FOE license and licenses for additional layered products, or you can choose the EOE or MCOE instead, depending on which layered products you want. For more information about the operating environments and their contents, refer to the *HP Operating Environments for OpenVMS Industry Standard 64 Version 8.2–1 for Integrity Servers SPD* at:

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

6.5 OpenVMS Version 8.2–1 Documentation

OpenVMS Version 8.2–1 documentation consists of two new manuals to supplement the current manuals, which were provided with OpenVMS Version 8.2. These new manuals provide the new information you need to install and use Version 8.2–1.

The following release documentation is included in the OpenVMS Version 8.2–1 media kit:

- Hardcopy documents
  - *HP OpenVMS Version 8.2–1 for Integrity Servers Upgrade and Installation Manual*
  - *HP OpenVMS Version 8.2–1 for Integrity Servers New Features and Release Notes*
  - *Cover Letter for HP OpenVMS Version 8.2–1 Integrity Servers*
  - *HP OpenVMS License Management Utility Manual*
  - *HP OpenVMS Version 8.2 Release Notes*
- Online documents
  - Online Documentation Library for HP OpenVMS I64 and Microsoft Windows Platforms CD

Not all books in a documentation set are revised for each release. Typically, each release contains a number of new and revised books, with the remainder of the set consisting of books from previous releases. For example, the OpenVMS Version 8.2 set includes several new and revised books, with the remaining titles being carryovers from previous releases. The books in each set are the most current versions available.

In addition, all the documentation is available from the HP OpenVMS Systems Documentation Web site:

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

Index

A

Adapters, 3–1
Associated product support, 3–2

B

Bitmap memory requirements for volume shadowing, 3–10
Booting, 1–7, 2–3
  delay, 1–8
BOOT_OPTIONS.COM, 1–8

C

Cell-based servers, 1–1
CLUSTER option, base addresses for images, 5–2
COLLECT command, 2–17
CREATE SERVICE command
  in InfoServer utility, 4–3

D

Debugger, 2–6
  See OpenVMS Debugger
  default data type, 2–7
  Heap Analyzer, 2–6
  new /START_LEVEL qualifier for SHOW STACK, 2–6
  overloaded symbol support in SHOW SYMBOL, 2–7
  preliminary Ada language support, 2–7
  SET MODULE command, 2–6
DECnet for OpenVMS, 1–3
DECnet-Plus for OpenVMS, 1–3
DECwindows Motif
  See HP DECwindows Motif
DECwindows X11 display server
  peripheral connection requirements, 1–9
DELETE SERVICE command
  in InfoServer utility, 4–7
Delta debugger
  available on I64, 2–7
Documentation, 6–5
Documentation corrections, 3–4
  $PUTMSG System Service, 3–10
  using IPC commands, 3–8

E

EFI driver, 3–1
EFI precautions, 3–10
EXIT command
  in InfoServer utility, 4–10

F

Fibre Channel
  performance data, 2–8
  system disk
    boot requirement, 1–8
Firmware
  for Integrity servers, 1–4
Freeware CD, 6–3

G

Granularity hints, 2–10
Graphics, 1–9

H

HELP command
  in InfoServer utility, 4–11
HP DECwindows Motif
  keyboard, 1–10
  startup messages, 1–9
  VGA console, 1–9

I

InfoServer utility
  commands
    CREATE SERVICE, 4–3
    DELETE SERVICE, 4–7
    EXIT, 4–10
    HELP, 4–11
    SAVE, 4–12
    SET SERVICE, 4–15
    SHOW SERVER, 4–18
    SHOW SERVICES, 4–19
    SHOW SESSIONS, 4–21
    SPAWN, 4–23
    START SERVER, 4–24
  exiting, 4–2

InfoServer utility (cont'd)
invoking, 4–1
Installation and upgrade information
networking options, 1–3
Integrity servers
firmware, 1–4
IPC Commands, 3–8

L

Layered products, 6–1
LIB$GET_CURR_INVO_CONTEXT
documentation correction, 3–7
LIB$GET_INVO_CONTEXT
documentation correction, 3–7
LIB$GET_INVO_HANDLE
documentation correction, 3–7
LIB$GET_PREV_INVO_CONTEXT
documentation correction, 3–7
LIB$GET_PREV_INVO_HANDLE
documentation correction, 3–7
LIB$GET_UIB_INFO
documentation correction, 3–7
LIB$PUT_INVO_REGISTERS
documentation correction, 3–7
LIBRARIAN
See Librarian Utility, 3–5
Librarian utility, 3–5
Library utility
corrected information
/accessing ELF object libraries, 3–6
/REMOVE, 3–5
Licenses, 1–2, 3–2, 6–3
Linker utility, 3–12, 5–1
conventions for image names, 5–18
images compiled with reduced floating–point, 5–11
linker map, 5–21
linking with ELF group symbols, 5–11
linking with UNIX-style weak symbols, 5–11
OpenVMS Alpha
handling of nonexistent files, 5–19
RMS_RELATED_CONTEXT option, 5–19
OpenVMS I64, 5–8
alignments for PSECT_ATTRIBUTE option, 5–19
flags set for /TRACEBACK, /DEBUG, /DSF, 5–6
handling initialized overlaid psects, 5–3
linking ELF common symbols selectively against shareable images, 5–6
linking relaxed reference/definition symbols, 5–6
restrictions for /SEGMENT_ATTRIBUTE, 5–13
understanding messages, 5–8

M

Media components, 6–1
Memory-resident sections
larger size, 2–10
Migration software, 1–6
Multipath Fibre Channel disk devices
managing list of boot devices, 1–8

N

Network
options, 1–3
update restrictions, 3–13
New Features
Debugger, 2–6

O

Object module name key, 3–11
OE DVD, 6–1
Open Source Tools CD, 6–3
OpenVMS Cluster systems
maximum number of OpenVMS I64 systems, 2–3
patch kits, 3–2
shared SCSI storage, 2–4
OpenVMS Debugger, 2–6, 3–4
Heap Analyzer conditions, 3–4
improved C++ support, 2–6
PC client documentation correction, 3–4
OpenVMS I64
booting from DVD, 1–7
OpenVMS I64 Boot Manager utility
removing devices, 1–8
scanning devices, 1–8
OpenVMS I64 Cluster system, 2–4
Operating Environment
See OE DVD, 6–1

P

Page sizes
larger, 2–10
Partition manager, 2–14
Patch kits for cluster compatibility, 3–2
Performance data
Fibre Channel configurations, 2–8
Power modes, 2–14
PPL units assignment tool, 2–15
Product
directories, 6–1
licensing, 6–5
Programming
$PUTMSG System Service correction, 3–10

PTD$READ routine
documentation clarification, 3–5

R

Reboot
automatic, 1–7
Remedial kits
obtaining, 1–3
RMI$_MODES item code, 3–8
Run-time library routines, 3–7
rx7620 server, 2–1
rx8620 server, 2–1

S

SAVE command
in InfoServer utility, 4–12
SCD
See System Code Debugger, 2–7
SDD
See System Dump Debugger, 2–7
Servers
rx7620, 2–1
rx8620, 2–1
Superdome, 2–1
SET SERVICE command
in InfoServer utility, 4–15
Shared SCSI storage
OpenVMS I64 two-node cluster, 2–4
SHOW CALL_FRAME parameter, 2–18
SHOW SERVER command
in InfoServer utility, 4–18
SHOW SERVICES command
in InfoServer utility, 4–19
SHOW SESSIONS command
in InfoServer utility, 4–21
SPAWN command
in InfoServer utility, 4–23
Starting address, 2–18
START SERVER command
in InfoServer utility, 4–24
Superdome server, 2–1
Support policy for software, 1–2
sx1000 Chipset, 2–1
System Code Debugger (SCD), 2–7
System disk
incompatibility with older systems, 1–4
System Dump Analyzer (SDA)
command
FC Performance, 2–8
utility, 2–15
System Dump Debugger (SDD), 2–7
System Event Log (SEL)
clearing on Integrity servers, 1–4
System management
invoking the InfoServer utility, 4–1

T

TCP/IP Services for OpenVMS, 1–3
TIE kit, 1–6
Traceback facility, 2–18
API problem fixed, 3–14
Translated Image Environment
See TIE kit, 1–6

V

Volume Shadowing for OpenVMS
EFI precautions with shadowed system disk, 3–10
memory requirements for bitmaps, 3–10