OpenVMS Version 7.0 New Features Manual

Order Number: AA–QSBFA–TE

December 1995

This manual describes the new features of the OpenVMS VAX Version 7.0 and the OpenVMS Alpha Version 7.0 operating systems.

Revision/Update Information: This is a new manual.

Software Version: OpenVMS Alpha Version 7.0
OpenVMS VAX Version 7.0

Digital Equipment Corporation
Maynard, Massachusetts

December 1995

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from Digital or an authorized sublicensor.

Digital conducts its business in a manner that conserves the environment and protects the safety and health of its employees, customers, and the community.

© Digital Equipment Corporation 1995. All rights reserved.

The following are trademarks of Digital Equipment Corporation: AlphaServer, AlphaStation, Bookreader, CI, DEC Ada, DEC Fortran, DEC VTX, DECchip, DECdirect, DECmcc, DECmigrate, DECnet, DECserver, DECterm, DECthreads, DECwindows, Digital, Digital UNIX, GIGAswitch HSC, HSJ, HSZ, InfoServer, LAT, MicroVAX, MSCP, OpenVMS, PATHWORKS, POLYCENTER, StorageWorks, TA, ThinWire, TMSCP, TURBOchannel, ULTRIX, VAX, VAX C, VAXcluster, VMS, VMScluster, VT, XMI, and the DIGITAL logo.

Motif is a registered trademark of Open Software Foundation, Inc.

NetView is a registered trademark of International Business Machines Corporation.

PostScript is a registered trademark of Adobe Systems Incorporated.

X/Open is a trademark of X/Open Company Limited.

Visual C++, Win32, Windows NT, and Windows 95 are registered trademarks of Microsoft Corporation.

All other trademarks and registered trademarks are the property of their respective holders.

ZK6457

This document is available on CD–ROM.

Contents

Preface xv

1 Summary of Version 7.0 New Features

2 General User Features

2.1 DCL Commands 2–1
2.1.1 New /ALLOCATION Qualifier for CREATE/DIRECTORY Command 2–1
2.1.2 New /ON Qualifier for RUN [process] Command 2–1
2.1.3 New Qualifiers for SET PROCESS Command (Alpha Only) 2–1
2.1.3.1 /CAPABILITIES Qualifier 2–2
2.1.3.2 /AFFINITY Qualifier 2–2
2.1.4 New /DEFAULT_CAPABILITIES Qualifier for START/CPU Command 2–3
2.1.5 New Display for SHOW CPU Command 2–3
2.2 High-Performance Sort/Merge Utility (Alpha Only) 2–4
2.3 Mail Utility 2–5
2.3.1 Support for Signature Files 2–6
2.3.1.1 Displaying Signature File Information 2–6
2.3.1.2 Including a Signature File by Default 2–6
2.3.1.3 Disabling the Default Setting 2–6
2.3.1.4 Overriding the Default Setting 2–7
2.3.1.5 Using Signature Files 2–7
2.3.2 /PAGE Qualifier Now Available with Mail Commands 2–8
2.3.3 /NEXT Qualifier Valid with SEARCH Command 2–9
2.4 Easy Internet Access 2–9
2.4.1 Internet Features Added to OpenVMS Systems 2–10

3 System Management Features

3.1 New and Changed System Parameters 3–1
3.1.1 System Parameters 3–1
3.1.1.1 CWCREPRC_ENABLE 3–1
3.1.1.2 DBGTK_SCRATCH (Alpha Only) 3–1
3.1.1.3 IO_PREFER_CPUS (Alpha Only) 3–1
3.1.1.4 MAXBOBMEM (Alpha Only) 3–2
3.1.1.5 MULTITHREAD (Alpha Only) 3–2
3.1.2 SPECIAL Parameters 3–2
3.1.2.1 FAST_PATH (Alpha Only) 3–2
3.1.3 Changed System Parameters 3–2
3.1.3.1 ACP_DATACHECK Has Three New Levels 3–2
3.2 LAT New Features 3–4
3.2.1 Displaying Speeds Greater than 57,600 Kbps 3–5

3.2.2 Using Multiple LAN Adapters . . . 3–5
3.2.2.1 Multiple LAN Addresses . . . 3–5
3.2.2.2 Supported Configurations . . . 3–5
3.2.2.3 Unsupported Configuration . . . 3–7
3.2.2.4 Creating Logical LAT Links . . . 3–7
3.2.2.5 Path Discovery . . . 3–8
3.2.3 Modifying LAT Parameters . . . 3–9
3.2.4 Managing Large Buffers . . . 3–9
3.2.5 Controlling Service Announcements . . . 3–11
3.2.6 Support for User-Written LAT Rating Algorithm . . . 3–11
3.2.7 New SET HOST/LAT Qualifier . . . 3–12
3.2.8 New LAT Item Codes . . . 3–12
3.3 Networking Support . . . 3–12
3.4 New Network Commands . . . 3–12
3.5 New Logical Names for OPCOM (Operator Communication Manager) . . . 3–12
3.6 Setting Correct Time Zone Information on Your System . . . 3–13
3.6.1 Understanding Time-Setting Concepts . . . 3–14
3.6.1.1 Coordinated Universal Time . . . 3–14
3.6.1.2 Time Differential Factor . . . 3–14
3.6.1.3 Daylight Saving Time and Standard Time . . . 3–15
3.6.1.4 Time Zones . . . 3–15
3.6.2 Determining Your System's Time Differential Factor . . . 3–15
3.6.3 Using UTC$TIME_SETUP.COM . . . 3–17
3.6.3.1 Setting the Time Zone on Your System . . . 3–17
3.6.3.2 Setting the TDF on Your System . . . 3–20
3.6.4 Adjusting for Daylight Saving Time and Standard Time . . . 3–21
3.6.5 Setting Time in a VMScluster Environment . . . 3–22
3.7 System Dump Analyzer (SDA) Features . . . 3–23
3.7.1 New SET FETCH Command . . . 3–23
3.7.2 Current Commands with New Features . . . 3–24
3.7.2.1 SHOW CLUSTER . . . 3–24
3.7.2.2 SHOW CONNECTIONS . . . 3–24
3.7.2.3 SHOW LAN . . . 3–24
3.7.2.4 SHOW LOCK . . . 3–24
3.7.2.5 SHOW PAGE_TABLE . . . 3–25
3.7.2.6 SHOW PFN_DATA . . . 3–25
3.7.2.7 SHOW PROCESS . . . 3–27
3.7.2.8 SHOW RESOURCE . . . 3–27
3.7.2.9 SHOW SUMMARY . . . 3–27
3.7.3 Display Changes . . . 3–28
3.7.4 Other Command Changes . . . 3–29
3.7.5 SDA VAX Dump File Process . . . 3–29
3.8 Warranted and Migration Support for VMScluster System Configurations . . . 3–30
3.9 Printing Features . . . 3–30
3.9.1 1024 Process Identifiers in Print Queuing Requests . . . 3–30
3.9.2 New Qualifier for Print Queues . . . 3–30

4 Programming Features

4.1 OpenVMS Alpha 64-Bit Addressing Support (Alpha Only) 4–1
4.2 OpenVMS Debugger 4–1
4.2.1 Debugging Optimized Code (Alpha Only) 4–1
4.2.2 Internationalization Features 4–3
4.2.3 SHOW CALLS and SHOW STACK Commands and Null Frame Procedures (Alpha Only) 4–4
4.2.4 CALL Command and Floating-Point Parameters 4–4
4.2.5 Customization Features for the DECwindows Motif Interface 4–4
4.2.6 Editor File Menu Items in the DECwindows Motif Interface 4–6
4.2.7 Command/Message View Popup Menu Items in the DECwindows Motif Interface 4–6
4.2.8 Documentation Changes 4–6
4.3 Heap Analyzer Support (Alpha Only) 4–6
4.4 DECthreads Features 4–7
4.4.1 DECthreads Implements Final POSIX 1003.1c Standard Style Interface 4–7
4.4.2 Thread Independent Services (TIS) Interface 4–7
4.5 DELTA/XDELTA Support for Debugging Multithreaded Applications (Alpha Only) 4–8
4.6 Global Section Limit Increased on OpenVMS Alpha Version 7.0 4–8
4.7 High-Performance Sort/Merge Utility—SOR$ Routines (Alpha Only) 4–8
4.8 Kernel Threads (Alpha Only) 4–10
4.8.1 Kernel Threads Advantages 4–10
4.8.2 New Kernel Threads Features 4–10
4.8.2.1 Multiple Execution Contexts Within a Process 4–10
4.8.2.2 Efficient Use of the OpenVMS and DECthreads Schedulers 4–11
4.9 Linking for Different Architectures 4–11
4.9.1 /ALPHA Qualifier 4–12
4.9.2 /VAX Qualifier 4–13
4.10 New /ALPHA Qualifier for Command Definition, Library, and Message Utilities 4–13
4.11 New LAT Item Codes (Alpha Only) 4–14
4.12 Mail Utility Features 4–14
4.12.1 Signature File User Profile Entry Field 4–14
4.12.2 Input Item Codes for the Signature File in the Send Context 4–15
4.12.3 Input Item Codes for the Signature File in the User Context 4–15
4.12.4 Output Item Code for the Signature File in the User Context 4–15
4.12.5 MAIL$SEND_BEGIN Routine Input Item Codes 4–15
4.12.5.1 MAIL$_SEND_SIGFILE and MAIL$_SEND_NO_SIGFILE 4–15
4.12.6 MAIL$USER_BEGIN and MAIL$USER_GET_INFO Routines Output Item Code 4–15
4.12.6.1 MAIL$_USER_SIGFILE 4–16
4.12.7 MAIL$USER_SET_INFO Routine Input Item Codes 4–16
4.12.7.1 MAIL$_USER_SET_SIGFILE and MAIL$_USER_SET_NO_SIGFILE 4–16
4.13 New STARLET Definitions for C (Alpha Only) 4–16
4.14 Spiralog Version 1.0 (Alpha Only) 4–18
4.15 Dump File Compression Features (Alpha Only) 4–18
4.15.1 Dump File Style 4–18
4.15.1.1 Controlling the Size of Page Files and Dump Files 4–20
4.15.1.2 Writing to the System Dump File 4–20
4.15.1.3 Writing to the System Page File 4–21

4.16 New SMBMSG$V_NO_INITIAL_FF Symbol for SMBMSG$K_PRINT_CONTROL Message Item Code 4–22
4.17 System Services 4–22
4.17.1 Fast IO System Services (Alpha Only) 4–22
4.17.2 New QIO Attribute, ATR$C_FILE_SYSTEM_INFO 4–23
4.17.3 New $CREPRC Argument 4–23
4.17.4 New System Services to Support CPU Scheduling (Alpha Only) 4–23
4.18 CPU Scheduling (Alpha Only) 4–24
4.18.1 Capabilities 4–24
4.18.1.1 User Capabilities 4–25
4.18.1.2 Scope of User Capabilities 4–25
4.18.1.3 Capability System Services 4–26
4.18.1.4 /CAPABILITIES Qualifier 4–26
4.18.2 Explicit Affinity 4–27
4.18.2.1 Scope of Explicit Affinity 4–27
4.18.2.2 Explicit Affinity System Service 4–28
4.18.2.3 /AFFINITY Qualifier 4–28
4.18.3 Implicit Affinity 4–28
4.18.4 Informational Services 4–29
4.18.4.1 DCL SHOW CPU 4–29
4.18.4.2 SDA SHOW PROCESS 4–29
4.18.4.3 $GETSYI - General System Information 4–29
4.18.4.4 $GETJPI - Kernel Thread Information 4–30
4.19 Alternative to Local Event Flags 4–30
4.20 Run-Time Library (RTL) Routines 4–31
4.20.1 Using LIB$CREATE_DIR to Create Large Directories 4–31
4.21 Wind/U Version 3.0 Run Time on OpenVMS Systems 4–31
4.22 ZIC Utility 4–31
4.22.1 Format 4–31
4.22.1.1 Parameters 4–32
4.22.1.2 Qualifiers 4–32
4.22.2 Description 4–32
4.22.2.1 Rule Lines 4–32
4.22.2.2 Zone Lines 4–34
4.22.2.3 Link Lines 4–35
4.22.3 Example 4–35

5 Optional Features for Improving I/O Performance

5.1 Fast I/O 5–1
5.1.1 Fast I/O Benefits 5–2
5.1.2 Using Buffer Objects 5–2
5.1.3 Differences Between Fast I/O Services and $QIO 5–3
5.1.4 Using Fast I/O Services 5–4
5.1.4.1 Using Fandles 5–4
5.1.4.2 Modifying Existing Applications 5–5
5.1.4.3 I/O Status Area (IOSA) 5–5
5.1.4.4 $IO_SETUP 5–6
5.1.4.5 $IO_PERFORM[W] 5–6
5.1.4.6 $IO_CLEANUP 5–6
5.1.4.7 Fast I/O FDT Routine (ACP_STD$FASTIO_BLOCK) 5–6
5.1.5 Additional Information 5–7
5.2 Fast Path 5–7
5.2.1 Fast Path Features and Benefits 5–7

5.2.2 Using Fast Path 5–8
5.2.3 Fast Path Restrictions 5–9

6 New DECamds Features

6.1 New Fields in the System Overview Window 6–2
6.2 Single Disk Summary Window 6–3
6.3 New Cluster Windows 6–5
6.3.1 Cluster Transition/Overview Summary Window 6–6
6.3.1.1 Data Displayed 6–7
6.3.1.2 Notes About Data Display 6–8
6.3.1.3 New Event in Window 6–8
6.3.1.4 From This Window... 6–9
6.3.2 SCA Summary Window 6–9
6.3.2.1 Notes About Data Display 6–11
6.3.2.2 New Event in Window 6–11
6.3.2.3 From This Window... 6–11
6.3.3 NISCA Summary Window 6–11
6.3.3.1 Data Displayed 6–13
6.3.3.2 Notes About Data Display 6–16

A New OpenVMS System Messages

A.1 List of Messages A–1

B DEC C XPG4 Localization Utilities—OpenVMS Version 6.2 Feature

B.1 GENCAT B–2
B.1.1 Format B–2
B.1.1.1 Parameters B–2
B.1.1.2 Qualifiers B–2
B.1.2 Description B–2
B.1.3 Errors B–4
B.1.4 Examples B–4
B.2 ICONV COMPILE B–6
B.2.1 Format B–6
B.2.1.1 Parameters B–6
B.2.1.2 Qualifier B–6
B.2.2 Description B–7
B.2.3 Errors B–10
B.2.4 Example B–10
B.3 ICONV CONVERT B–10
B.3.1 Format B–10
B.3.1.1 Parameters B–10
B.3.1.2 Qualifiers B–10
B.3.2 Description B–11
B.3.3 Example B–11
B.4 LOCALE COMPILE B–11
B.4.1 Format B–11
B.4.1.1 Parameter B–11
B.4.1.2 Qualifiers B–12
B.4.2 Description B–13
B.4.3 Errors B–13
B.4.4 Example B–14

B.5 LOCALE LOAD B–14
B.5.1 Format B–14
B.5.1.1 Parameter B–14
B.5.1.2 Qualifiers B–14
B.5.2 Description B–15
B.6 LOCALE UNLOAD B–15
B.6.1 Format B–15
B.6.1.1 Parameter B–15
B.6.1.2 Qualifiers B–15
B.6.2 Description B–15
B.7 LOCALE SHOW CHARACTER_DEFINITIONS B–15
B.7.1 Format B–15
B.7.1.1 Parameters B–16
B.7.1.2 Qualifiers B–16
B.7.2 Description B–16
B.7.3 Example B–16
B.8 LOCALE SHOW CURRENT B–16
B.8.1 Format B–16
B.8.1.1 Parameters B–16
B.8.1.2 Qualifiers B–16
B.8.2 Description B–16
B.8.3 Example B–17
B.8.4 Errors B–18
B.9 LOCALE SHOW PUBLIC B–18
B.9.1 Format B–18
B.9.1.1 Parameters B–18
B.9.1.2 Qualifiers B–18
B.9.2 Description B–18
B.9.3 Example B–18
B.10 LOCALE SHOW VALUE B–18
B.10.1 Format B–18
B.10.1.1 Parameter B–19
B.10.1.2 Qualifiers B–21
B.10.2 Errors B–21
B.10.3 Description B–21
B.10.4 Examples B–21
B.11 Locale File Format B–22
B.11.1 Locale Categories B–22
B.11.1.1 Overriding Defaults B–22
B.11.1.2 Category Source Definitions B–23
B.11.2 LC_COLLATE Category B–23
B.11.2.1 The collating-element Statement B–24
B.11.2.2 The collating-symbol Statement B–25
B.11.2.3 The order_start Statement B–25
B.11.3 LC_CTYPE Category B–27
B.11.4 LC_MESSAGES Category B–30
B.11.5 LC_MONETARY Category B–31
B.11.5.1 LC_MONETARY Keywords B–31
B.11.5.2 Monetary Format Variations B–34
B.11.6 LC_NUMERIC Category B–35
B.11.7 LC_TIME Category B–36
B.11.7.1 Keywords B–37
B.11.7.2 Field Descriptors B–39
B.11.7.3 Sample Locale Definition B–41

B.12 Character Set Description (Charmap) File B–42
B.12.1 Portable Character Set B–42
B.12.2 Components of a Charmap File B–45

C SCSI as a VMScluster Storage Interconnect—OpenVMS Alpha Version 6.2 Feature

C.1 Conventions Used in This Appendix C–1
C.1.1 SCSI ANSI Standard C–2
C.1.2 Symbols Used in Figures C–2
C.2 Accessing SCSI Storage C–2
C.2.1 Single-Host SCSI Access in VMScluster Systems C–2
C.2.2 Multiple-Host SCSI Access in VMScluster Systems C–3
C.3 Configuration Requirements and Hardware Support C–4
C.3.1 Configuration Requirements C–4
C.3.2 Hardware Support C–5
C.4 SCSI Interconnect Concepts C–6
C.4.1 Number of Devices C–6
C.4.2 Performance C–6
C.4.3 Distance C–7
C.4.4 Cabling and Termination C–8
C.5 SCSI VMScluster Hardware Configurations C–9
C.5.1 Systems Using Add-On SCSI Adapters C–9
C.5.1.1 Building a Basic System Using Add-On SCSI Adapters C–10
C.5.1.2 Building a System That Allows a Server to Be Removed (Using DWZZA Converters) C–11
C.5.1.3 Building a System That Allows Additional Features and Performance Using an HSZ40 Controller C–12
C.5.1.4 Building a System with More Enclosures or Greater Separation C–14
C.5.2 Building a System Using Internal SCSI Adapters C–18
C.6 Installation C–19
C.6.1 Step 1: Meet SCSI Grounding Requirements C–20
C.6.2 Step 2: Configure SCSI Node IDs C–20
C.6.2.1 Configuring Device IDs on Multiple-Host SCSI Buses C–21
C.6.2.2 Configuring Device IDs on Single-Host SCSI Buses C–22
C.6.3 Step 3: Power Up and Verify SCSI Devices C–22
C.6.4 Step 4: Show and Set SCSI Console Parameters C–24
C.6.5 Step 5: Install the OpenVMS Operating System C–26
C.6.6 Step 6: Configure Additional Systems C–26
C.7 Supplementary Information C–26
C.7.1 Running the CLUSTER_CONFIG Command Procedure C–26
C.7.2 Error Reports and OPCOM Messages in Multiple-Host SCSI Environments C–28
C.7.2.1 SCSI Bus Resets C–28
C.7.2.2 SCSI Timeouts C–29
C.7.2.3 Mount Verify C–29
C.7.2.4 Shadow Volume Processing C–30
C.7.2.5 Expected OPCOM Messages in Multiple-Host SCSI Environments C–30
C.7.2.6 Error-Log Basics C–30
C.7.2.7 Error-Log Entries in Multiple-Host SCSI Environments C–31
C.7.3 Restrictions and Known Problems C–32

C.7.4 Troubleshooting C–33
C.7.4.1 Termination Problems C–33
C.7.4.2 Booting or Mounting Failures Caused by Incorrect Configurations C–33
C.7.4.2.1 Bugchecks During the Bootstrap Process C–33
C.7.4.2.2 Mount Failures C–35
C.7.4.3 Grounding C–36
C.7.4.4 Interconnect Lengths C–36
C.7.5 SCSI Arbitration Considerations C–36
C.7.5.1 Arbitration Issues in Multi-Disk Environments C–36
C.7.5.2 Solutions for Resolving Arbitration Problems C–37
C.7.5.3 Arbitration and Bus Isolators C–37
C.7.6 Removal and Insertion of SCSI Devices While the VMScluster System is Operating C–38
C.7.6.1 Terminology for Describing Hot Plugging C–38
C.7.6.2 Rules for Hot Plugging C–39
C.7.6.3 Procedures for Ensuring That a Device or Segment Is Inactive . . . C–41
C.7.6.4 Procedure for Hot Plugging StorageWorks SBB Disks C–42
C.7.6.5 Procedure for Hot Plugging HSZ40s C–43
C.7.6.6 Procedure for Hot Plugging Host Adapters C–44
C.7.6.7 Procedure for Hot Plugging DWZZAs C–44
C.7.7 OpenVMS Requirements for Devices Used on Multiple-Host SCSI VMScluster Systems C–46
C.7.8 Grounding Requirements C–47

D VMScluster Systems That Span Multiple Sites—OpenVMS Version 6.2 Feature

D.1 What Is a Multiple-Site VMScluster System? D–1
D.1.1 ATM, DS3, and FDDI Intersite Links D–2
D.1.2 Benefits of Multiple-Site VMScluster Systems D–3
D.1.3 General Configuration Guidelines D–4
D.2 Using FDDI to Configure Multiple-Site VMScluster Systems D–4
D.3 Using WAN services to Configure Multiple-Site VMScluster Systems D–5
D.3.1 The ATM Communications Service D–5
D.3.2 The DS3 Communications Service D–6
D.3.3 FDDI-to-WAN Bridges D–6
D.3.4 Guidelines for Configuring ATM and DS3 in a VMScluster System . . . D–7
D.3.4.1 Requirements D–7
D.3.4.2 Recommendations D–8
D.3.5 Availability Considerations D–10
D.3.6 Specifications D–10
D.4 Managing VMScluster Systems Across Multiple Sites D–12
D.4.1 Methods and Tools D–13
D.4.2 Shadowing Data D–14
D.4.3 Monitoring Performance D–14

E Other OpenVMS Version 6.2 New Features

E.1 VMScluster Systems New Features E–1
E.1.1 OpenVMS Cluster Client Software E–1
E.1.2 Support for TMSCP Served SCSI Tapes E–1
E.1.2.1 No TMSCP Server Support for SCSI Retension Command E–2
E.1.3 Enhanced Support for HSJ, HSC, and HSD Series Controller Failover E–2

Index

Examples

C–1 Adding a Node to a SCSI Cluster C–27

Figures

3–1 Multiple Address LAT Configuration: One LAN with Mixed Version LAT Nodes 3–5
3–2 Multiple Address LAT Configuration: Two LANs with Mixed Version LAT Nodes 3–6
3–3 Multiple Address LAT Configuration: Two LANs with Version 5.3 LAT Nodes 3–7
3–4 Unsupported Multiple Address LAT Configuration 3–7
3–5 LAT FDDI Ring and Large Buffers 3–10
3–6 Time Differential Factor Map 3–16
3–7 VMScluster Version Pairings 3–30
6–1 DECamds Data Window Hierarchy 6–2
6–2 System Overview Window 6–3
6–3 Single Disk Summary Window 6–4
6–4 Cluster Transition/Overview Summary Window 6–6
6–5 SCA Summary Window 6–10
6–6 NISCA Summary Window 6–12
C–1 Conventions: Key to Symbols Used in Figures C–2
C–2 Highly Available Servers for Shared SCSI Access C–3
C–3 Maximum Stub Lengths C–9
C–4 Conceptual View: Basic SCSI System C–10
C–5 Sample Configuration: Basic SCSI System Using AlphaServer 1000, KZPAA Adapter, and BA350 Enclosure C–11
C–6 Conceptual View: SCSI System with Bus Isolator (DWZZA Converter) C–12
C–7 Sample Configuration: SCSI System with DWZZA Converter, AlphaServer 1000 Systems, and BA350 Enclosure C–12
C–8 Conceptual View: System Using Differential Controllers C–13
C–9 Sample Configuration: System Using HSZ40 Controller in an SW300 Enclosure C–14
C–10 Conceptual View: Using DWZZAs to Allow for Increased Separation or More Enclosures C–15

C–11 Sample Configuration: Using DWZZAs to Allow for Increased Separation or More Enclosures C–16
C–12 Sample Configuration: Three Hosts on a SCSI Bus C–17
C–13 Conceptual View: SCSI VMScluster System Using Internal Adapters C–18
C–14 Sample Configuration: SCSI VMScluster System with AlphaStation 200 Systems Using Internal Adapters C–19
C–15 Setting Allocation Classes for SCSI Access C–21
C–16 SCSI Bus Topology C–39
C–17 Hot Plugging a Bus Isolator C–41
D–1 Site-to-Site Link Between Philadelphia and Washington D–2
D–2 Multiple-Site VMScluster Configuration with Remote Satellites D–4
D–3 ATM/SONET OC-3 Service D–6
D–4 DS3 Service D–6
D–5 Multiple-Site VMScluster Configuration Connected by DS3 . . . . . D–7

Tables

1–1 Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features 1–1
2–1 High-Performance Sort/Merge: Differences in Behavior 2–5
3–1 LAT$RATING Sources 3–11
3–2 Time Zone Acronyms 3–18
3–3 Page Frame Number Information—Line One Fields 3–25
3–4 Page Frame Number Information—Line Two Fields 3–26
3–5 Display Changes 3–28
4–1 High-Performance Sort/Merge: Differences in SOR$ Routine Behavior 4–9
4–2 Logical Names for Cross-Architecture Linking 4–11
4–3 LAT Node Entity Item Codes 4–14
4–4 LAT Port Entity Item Code 4–14
4–5 Structures Used by _NEW_STARLET Prototypes 4–17
4–6 DUMPSTYLE Mask 4–19
4–7 Comparison of Full and Selective Dump Files 4–20
4–8 SMBMSG$V_NO_INITIAL_FF Symbol 4–22
4–9 $GETSYI item codes for CPU Scheduling 4–30
4–10 $GETJPI item codes for CPU Scheduling 4–30
4–11 Day the Rule Becomes Effective 4–33
4–12 Time of Day the Rule Becomes Effective 4–33
6–1 New Fields in the System Overview Window 6–3
6–2 Data Items in the Single Disk Summary Window 6–5
6–3 Data Items in the Summary Panel of the Cluster Transition/Overview Summary Window 6–7
6–4 Data Items in the Cluster Members Panel of the Cluster Transition/Overview Summary Window 6–8
6–5 Data Items in the SCA Summary Window 6–10
6–6 Data Items in the Transmit Panel 6–13
6–7 Data Items in the Receive Panel 6–13

6–8	Data Items in the Congestion Control Panel	6–14
6–9	Data Items in the Channel Selection Panel	6–15
6–10	Data Items in the VC Closures Panel	6–15
6–11	Data Items in the Packets Discarded Panel	6–15
B–1	Special Characters	B–4
B–2	Codeset Declarations	B–7
B–3	Locale Categories and Keywords	B–19
B–4	LC_COLLATE Category Keywords	B–24
B–5	LC_CTYPE Category Keywords	B–28
B–6	LC_MESSAGES Category Keywords	B–31
B–7	LC_MONETARY Category Keywords	B–32
B–8	Monetary Format Variations	B–34
B–9	LC_NUMERIC Category Keywords	B–36
B–10	LC_TIME Category Keywords	B–37
B–11	LC_TIME Locale Field Descriptors	B–39
B–12	Portable Character Set	B–42
C–1	Requirements for SCSI VMScluster Configurations	C–4
C–2	Supported Hardware for SCSI VMScluster Systems	C–5
C–3	Maximum Data Transfer Rates in Megabytes per Second	C–7
C–4	Maximum SCSI Interconnect Distances	C–8
C–5	Internal SCSI Cable Lengths	C–19
C–6	Steps for Installing a SCSI VMScluster System	C–20
C–7	SCSI Environment Parameters	C–24
C–8	Steps for Installing Additional Nodes	C–26
C–9	Steps for Ensuring Proper Grounding	C–47
D–1	DS3 Protocol Options	D–8
D–2	Bellcore and VMScluster Requirements and Goals Terminology	D–11
D–3	VMScluster DS3 & SONET OC3 Error Performance Requirements	D–12
E–1	TMSCP_SERVE_ALL System Parameter Settings	E–2

Preface

Intended Audience

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

This document contains descriptions of the new features for Version 7.0 of the OpenVMS VAX and OpenVMS Alpha operating systems. For information about how some of the new features might affect your system, read the *OpenVMS Version 7.0 Release Notes* before you install, upgrade, or use Version 7.0.

Document Structure

This manual is organized as follows:

- Chapter 1 contains a summary of the new OpenVMS software features.
- Chapter 2 describes new features of interest to general users of the OpenVMS VAX and OpenVMS Alpha operating systems.
- Chapter 3 describes new features that are applicable to the tasks performed by system managers.
- Chapter 4 describes new features that support programming tasks.
- Chapter 5 describes new optional features for improving I/O performance.
- Chapter 6 describes new features of DECamds.
- Appendix A describes new or changed messages from the Help Message database.
- Appendix B describes the new DEC C XPG4 localization utilities. This is an OpenVMS Version 6.2 new feature that is not published elsewhere in the printed documentation.
- Appendix C describes how VMScluster systems support the Small Computer Systems Interface (SCSI) as a storage interconnect. This is an OpenVMS Version 6.2 new feature that is not published elsewhere in the printed documentation.
- Appendix D discusses multiple-site VMScluster configurations, with an emphasis on the new wide area network ATM and DS3 communications services. This is an OpenVMS Version 6.2 new feature that is not published elsewhere in the printed documentation.
- Appendix E describes other OpenVMS Version 6.2 new features that have not yet been published in the printed documentation.

Related Documents

Refer to the following documents for detailed information about the software features described in this manual:

- *DEC C Run-Time Library Reference Manual for OpenVMS Systems*
- *OpenVMS DCL Dictionary*
- *OpenVMS RTL Library (LIB$) Manual*
- *OpenVMS System Management Utilities Reference Manual*
- *OpenVMS System Manager's Manual*
- *OpenVMS System Messages: Companion Guide for Help Message Users*
- *OpenVMS System Services Reference Manual*
- *New OpenVMS Alpha System Services Manual*
- *OpenVMS User's Manual*
- *OpenVMS VAX Version 7.0 Upgrade and Installation Manual*
- *OpenVMS Version 7.0 Release Notes*
- *VMScluster Systems for OpenVMS*

How To Order Additional Documentation

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

Telephone and Direct Mail Orders

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

ZK–7654A–GE

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

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

Reader's Comments

Digital welcomes your comments on this manual.

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

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

Conventions

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

The following conventions are used to identify information specific to OpenVMS Alpha or to OpenVMS VAX:

Alpha	The Alpha icon denotes the beginning of information specific to OpenVMS Alpha.
VAX	The VAX icon denotes the beginning of information specific to OpenVMS VAX.
♦	The diamond symbol denotes the end of a section of information specific to OpenVMS Alpha or to OpenVMS VAX.

In this manual, every use of DECwindows and DECwindows Motif refers to DECwindows Motif for OpenVMS software.

The following conventions are also used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)

. . .	Horizontal ellipsis points in examples indicate one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	Vertical ellipsis points indicate the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In command format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in an OpenVMS file specification or in the syntax of a substring specification in an assignment statement.)
{ }	In command format descriptions, braces indicate a required choice of options; you must choose one of the options listed.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in Bookreader versions of the manual.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system messages (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *device-name* contains up to five alphanumeric characters).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`struct`	Monospace type in text identifies the following C programming language elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1

Summary of Version 7.0 New Features

Table 1–1 summarizes the new software features supported by OpenVMS Alpha and OpenVMS VAX Version 7.0.

Table 1–1 Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features

General User Features	
DCL commands	New /ALLOCATION=*n* qualifier to the CREATE/DIRECTORY command.
	New /NO_INITIAL_FF qualifier to the INITIALIZE/QUEUE, SET QUEUE, and START QUEUE commands.
	New /ON qualifier for RUN [process] command.
	New SET PROCESS command qualifiers and new START/CPU /DEFAULT_CAPABILITIES command (Alpha systems only).
	New display for SHOW CPU command (Alpha systems only).
High-performance Sort/Merge utility	Command interface to the optional high-performance Sort/Merge utility.
Mail utility	Support for signature files, /PAGE qualifier, and /NEXT qualifier (with SEARCH command).
Internet access	Support for commercial Internet products and abundant freeware through an OpenVMS Web server.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features

System Management Features	
System parameters	New parameters: • CWCREPRC_ENABLE • DBGTK_SCRATCH (Alpha systems only) • IO_PREFER_CPUS • MAXBOBMEM (Alpha systems only) • MULTITHREAD (Alpha systems only) • FAST_PATH (Alpha systems only) • SCSICLUSTER_P[1-4] (Alpha systems only) Changed parameter: • ACP_DATACHECK
LAT software	Support for greater terminal speeds, multiple LAN adapters, large buffers, disabling service announcements, user-written LAT rating algorithm, and a new SET HOST/LAT qualifier (/FRAME).
Networking support	Support for network products, such as DECnet Phase IV, during installation.
New network commands	Allow system managers to view more information about and control various network services through a revised SHOW NETWORK command.
Local time zone support	Supports the DEC C RTL implementation of its date/time support using a Universal Coordinated Time (UTC) model.
OPCOM	New logicals that regulate the flow of OPCOM messages.
System dump analyzer (SDA)	New SDA command and changes to existing commands for this release.
SDA VAX Dump File Process	Supports RMS file access for processing dump files.
VMScluster systems migration	Support for mixed-version and mixed-architecture VMSclusters.
Printing features	Greater print queue support for users with a large number of process identifiers. Also, provide control over form feed during printing with the /NO_INITIAL_FF qualifier.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features

Programming Features	
OpenVMS Alpha 64-bit virtual addressing	Allows per-process virtual addressing for accessing dynamically mapped data beyond 32-bit limits.
Debugger functionality with optimized code	On OpenVMS Alpha systems, the ability to debug code compiled with full optimization has been significantly improved.
Debugger internationalization features	Provide for multibyte characters, Japanese user-defined words and identifiers, and wide character strings.
Debugger SHOW CALLS command and null frame procedures	The display resulting from a SHOW CALLS command includes the frames of null frame procedures.
Debugger CALL command and floating-point parameters	The CALL command now allows you to pass floating-point parameters by value, on OpenVMS Alpha systems as well as OpenVMS VAX systems.
Debugger customization features for Motif interface	Improved user customization features are provided by the VMSDEBUG.DAT resource file and new menu items.
Debugger Editor File menu	Two new menu items, Refresh File and Close File, have been added.
Heap Analyzer support	The Heap Analyzer, previously available only on OpenVMS VAX systems, is now available on OpenVMS Alpha systems.
DECthreads	Provides an implementation of the final POSIX 1003.1c Standard Style Interface.
Thread Independent Services (TIS) interface	DECthreads implements TIS, which provides services that assist with the development of thread-safe libraries.
DELTA/XDELTA support for threads (Alpha systems only)	Thread ID displayed at the breakpoint if the process is multithreaded.
Global Section Limit (Alpha systems only)	Limit is now increased to 65,535.
High-performance Sort/Merge utility (Alpha systems only)	SOR$ routines callable interface to the optional high-performance Sort /Merge utility.
Kernel threads (Alpha systems only)	Process can contain an address space wherein a single thread or multiple threads execute concurrently.
Linker utility	Allows you to create OpenVMS Alpha images on an OpenVMS VAX system and OpenVMS VAX images on an OpenVMS Alpha system.
Command Definition, Library, and Message utilities	A new /ALPHA qualifier has been added, and the behavior of the /VAX qualifier has been changed.
New LAT item codes (Alpha systems only)	Node and port entities have several new item codes.
Mail utility	Provides a new signature file user profile entry field, new item codes to implement the signature file, new item codes for the Mail utility, and new item codes for Mail utility routines.
New STARLET definitions for C	Provide C function prototype definitions for system services, as well as new and enhanced data structure definitions.
Spiralog Version 1.0 (Alpha systems only)	Included in the OpenVMS license, Spiralog is an option that increases data write performance and provides quicker backup rates.
SDA dump file compression (Alpha systems only)	OpenVMS Alpha supports dump file compression features.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features

Programming Features	
New SMBMSG$V_NO_INITIAL_FF symbol for SMBMSG$K_PRINT_CONTROL message item code of SMB$READ_MESSAGE_ITEM routine	The SMB$READ_MESSAGE_ITEM routine's SMBMBG$K_PRINT_CONTROL message item code has a new SMBMSG$V_NO_INITIAL_FF symbol.
New QIO attribute, ATR$C_FILE_SYSTEM_INFO	Checks which file system created a file or directory.
New node argument for $CREPRC	Allows creation of a detached process on another node.
$CPU_CAPABILITIES system service (Alpha systems only)	Allows modification of the user capability set for a specified CPU, or for the global CPU default.
$PROCESS_CAPABILITIES system service (Alpha systems only)	Allows modification of the user capability set for a specified process thread, or for the global process default.
$PROCESS_AFFINITY system service (Alpha systems only)	Allows modification of the CPU affinity set for a specified process thread.
$SET_IMPLICIT_AFFINITY system service (Alpha systems only)	Controls or retrieves the activation state of the implicit affinity capability for a specified process thread, or for the global process default.
New Local Event Flag EFN 128 (ENF$C_ENF)	Used with the wait forms of system services, such as SYS$QIOW, SYS$ENQW, or SYS$SYNCH. ENF$C_ENF does not need to be initialized, nor does it need to be reserved or freed.
New initial-allocation argument for the LIB$CREATE_DIR RTL	Specifies the initial number of blocks to be allocated to the directory.
Wind/U Version 3.0 Run Time	OpenVMS 7.0 now includes the Wind/U Version 3.0 run-time binaries.
ZIC utility	Time zone compiler to create time zone source files.
Optional Features for Improving I/O Performance	
Fast I/O	Provides a speedy alternative to the $QIO system service. This new set of services are intended as a substitute for the subset of $QIO operations that deal with high-volume read/write requests.
Fast Path	Improves overall I/O performance. Fast Path restructures and optimizes class and port device driver code around high-volume code paths, which creates a streamlined device path.
DECamds Features	
New fields in the System Overview window	Number of Processes in CPU Queues field, Operating System Version field, and Hardware Model field.
Single Disk Summary window	Provides summary data about each node in the group in which a disk is available.
Three new cluster windows	Cluster Transition/Overview Summary window, System Communication Summary (SCA) window, and Network Interconnect System Communication Architecture (NISCA) Summary window.

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features

OpenVMS System Messages	
New and changed system messages	New and revised messages for the following utilities: • System Bugcheck (BUGCHECK) • LAT facility (LAT) • Linker utility (LINK) • NETWRK, SET/SHOW/START/STOP NETWORK commands • System Dump Analyzer (SDA) • SET command and utility • System Services (SYSTEM)
DEC C XPG4 Localization Utilities	
GENCAT	Merges one or more message text source files into a message catalog file.
ICONV COMPILE	Creates a conversion table file from a conversion table source file.
INCONV CONVERT	Converts the characters in a file from one codset to another.
LOCALE COMPILE	Converts a locale source file into a binary locale file.
LOCALE LOAD	Loads the specified locale name into memory as a shared, read-only global data item.
LOCALE UNLOAD	Unloads the specified locale name from memory.
LOCALE SHOW CHARACTER_ DEFINITIONS	Lists character set description files (charmaps).
LOCALE SHOW CURRENT	Displays a summary of the current international environment.
LOCALE SHOW PUBLIC	Lists all the public locales on the system.
LOCALE SHOW VALUE	Displays the value of one or more keywords from the current international environment.
Locale File Format	Describes the standard, supported locale categories and lists any overriding defaults and category source definitions.
Character Set Description	Lists and describes the components of the character description file (charmap).

(continued on next page)

Table 1–1 (Cont.) Summary of OpenVMS VAX and OpenVMS Alpha Version 7.0 Software Features

SCSI Storage Interconnect Feature	
Accessing Storage	Access SCSI devices by CI interconnect, Digital Storge System Interconnect (DSSI), and SCSI adapter.
Configuration Requirements	The basic components needed to configure a SCSI system.
SCSI Interconnect Concepts	The rules that govern the interactions between initiators and SCSI targets.
Hardware Configurations	Illustrates the various hardware configurations that are available.
Installation	How to install a SCSI system.
Supplementary Information	Additional technical information such as, how to add nodes and interpret error reports.
Restrictions	List of known probems and restrictions.
Troubleshooting	Troubleshooting tips for common problems in a VMScluster system that uses a SCSI interconnect.
Arbitration Considerations	How to control the SCSI bus and achieve the optimum results.
Removal and Insertion of SCSI Devices	How to remove or insert a SCSI device during system operation.
OpenVMS Requirements	The requirements for devices used on a multiple-host SCSI VMScluster system.
Grounding Requirements	The grounding requirements for electrical systems in a SCSI VMScluster.
Multi-Site VMScluster Systems	
Using FDDI	The most common method to connect two distant VMScluster sites.
Using WAN	How to bridge an FDDI interconnect to the ATM or DS3 communications services and provides guidelines for using these services to configure multi-site VMScluster systems.
Managing Multi-Site Systems	Considerations and system management tools and techniques.
OpenVMS Version 6.2 Features	
OpenVMS Cluster client software	New license type that provides a low-cost cluster client product for Alpha and VAX workstations.
Support for TMSCP-served SCSI tapes	Enhanced TMSCP server that makes the following types of tapes locally connected across a cluster: • TA series for CI • TF series for DSSI • TZ and TLZ tapes for SCSI
Enhanced Support for HSJ, HSC, and HSD Series Controller Failover	Connect dual-ported disks to pairs of controllers that are attached to different star couplers.

2
General User Features

This chapter provides new features information for all users of the OpenVMS operating system.

2.1 DCL Commands

This section describes new and updated DCL commands.

2.1.1 New /ALLOCATION Qualifier for CREATE/DIRECTORY Command

The CREATE/DIRECTORY command has a new qualifier, /ALLOCATION=n, where *n* specifies the initial number of blocks to be allocated to each of the specified directories.

This qualifier applies only to Files–11 Level 2 volumes; it is ignored for other volumes. The default allocation is 1 block.

This qualifier is useful for creating large directories, for example MAIL.DIR;1. It can improve performance by avoiding the need for later dynamic expansion of the directory.

2.1.2 New /ON Qualifier for RUN [process] Command

The new RUN [process] command qualifier /ON allows a user to specify on what VMScluster node a process is to be created. The qualifier value is a 1- to 6-character string containing the SCS node name of that node.

For example, to create a process named BAR on node FOO that runs MY_PROG.EXE, enter the following command:

```
$ RUN $10$DKB100:[SMITH]MY_PROG.EXE /DETACH /ON="FOO"/PROCESS_NAME="BAR"
```

Note that the disk containing the image must be mounted on the specified node. Because the disk might not be mounted on the node on which the command is entered, the RUN command processor does not check whether the image exists. Consequently, the command can complete without error even though the created process aborts immediately because the image file cannot be found.

2.1.3 New Qualifiers for SET PROCESS Command (Alpha Only)

Alpha

The following sections describe new qualifiers for the SET PROCESS command.

2.1.3.1 /CAPABILITIES Qualifier

SET PROCESS /[NO]CAPABILITIES[/SET=(n)][/CLEAR=(n)][/PERMANENT]

The /CAPABILITIES command qualifier allows bits in the user process capability mask to be set or cleared individually, in groups, or all at once. This qualifier is mutually exclusive with the /AFFINITY qualifier described in the next section.

The /NOCAPABILITIES qualifier clears all user capability bits currently set, based on the setting of the /PERMANENT qualifier. Specifying /CAPABILITIES itself has no direct effect other than to indicate the target of the operations specified by the following secondary qualifiers:

/SET=(n,[,...])	Sets all user capabilities defined by the position values n, where n has the range of 1 to 16.
/CLEAR=(n,[,...])	Clears all user capabilities defined by the position values n, where n has the range of 1 to 16.
/PERMANENT	Forces the operations to be performed on the permanent user mask as well as the current, effectively making the changes permanent for the life of the thread or process. (The default behavior is to affect only the running image copy of the capabilities.)

The secondary qualifiers can all be used at once as long as the set of user capability bits defined in the /SET and /CLEAR parameters does not overlap.

The privileges required to execute this command match those required by the $PROCESS_AFFINITY system service. ALTPRI is the base privilege required to make any modifications, and the only privilege required to modify the current owner's process/thread. To make modifications in the same UIC group, GROUP is required. Otherwise, to make modifications to any unrelated process/thread, WORLD privilege is required.

As with the other SET PROCESS command qualifiers, the bit operations occur on the current process if no /IDENTIFICATION qualifier or explicit process name parameter is specified. Note that the /IDENTIFICATION qualifier allows this command to affect individual Kernel Thread PIDs; since each KTB is a separate runnable entity, these commands treat them as discrete entities in terms of capabilities. Specifying a process name does not imply that all threads associated with the process are affected; the SET PROCESS command affects only the initial thread of a multithreaded process.

2.1.3.2 /AFFINITY Qualifier

/[NO]AFFINITY[/SET=(n)][/CLEAR=(n)][/PERMANENT]

The /AFFINITY qualifier allows bits in the affinity mask to be set or cleared individually, in groups, or all at once. This qualifier is mutually exclusive with the /CAPABILITIES qualifier.

The /NOAFFINITY qualifier clears all affinity bits currently set, based on the setting of the /PERMANENT qualifier. Specifying /AFFINITY itself has no direct effect other than to indicate the target of the operations specified by the following secondary parameters:

/SET=(n,[,...])	Sets all CPU affinities defined by the position n, where n has a range of 1 to 32 and is restricted to the set of currently active CPUs.
/CLEAR=(n,[,...])	Clears all CPU affinities defined by the position values n, where n has a range of 1 to 32 and is restricted to the set of currently active CPUs.

/PERMANENT — Forces the operations to be performed on the permanent user mask as well as the current, effectively making the changes permanent for the life of the thread or process. (The default behavior is to affect only the running image copy of the affinities.)

The secondary qualifiers can all be used at once as long as the set of affinity bits defined in the /SET and /CLEAR parameters do not overlap.

The privileges required to execute this command match those required by the $PROCESS_AFFINITY system service. ALTPRI is the base privilege required to make any modifications, and the only privilege required to modify the current owner's process/thread. To make modifications in the same UIC group, GROUP is required. Otherwise, to make modifications to any unrelated process/thread, WORLD privilege is required.

As with the other SET PROCESS command qualifiers, the bit operations occur on the current process if no /IDENTIFICATION qualifier or explicit process name parameter is specified. Note that the /IDENTIFICATION qualifier allows this command to affect individual Kernel Thread PIDs; since each KTB is a separate runnable entity, these commands treat them as discrete entities in terms of affinities. Specifying a process name does not imply that all threads associated with the process are affected; the SET PROCESS command affects only the initial thread of a multithreaded process.♦

2.1.4 New /DEFAULT_CAPABILITIES Qualifier for START/CPU Command

The new /DEFAULT_CAPABILITIES qualifier for the START/CPU command eliminates all previous capability (user and system) modifications for the specified CPU and reinitializes them with the values in the global initialization variable SCH$GL_DEFAULT_CPU_CAP.

Normally, user capabilities survive CPU shutdowns and restarts (not reboots), making the downtime as transparent to the user as possible. The CPU user capability bits are only initialized from SCH$GL_DEFAULT_CPU_CAP at the first boot of the CPU. (The system capability bits, however, are reinitialized to their defaults taken from SCH$GL_DEFAULT_CPU_CAP.)

There might be times, however, when the CPU needs to be returned to a known, consistent state. You can use the /DEFAULT_CAPABILITIES qualifier to mimic the behavior of the initial bootstrap of the CPU.

2.1.5 New Display for SHOW CPU Command

The display portion of the SHOW CPU command is changed to distinguish between CPU affinity due to system capabilities, user capabilities, and process affinity. Since the code currently recognizes the longwords these entities reside in, the extent of changes is cosmetic. The new code will present the system and user capabilities for every CPU as well as for every thread that has values differing from the default process value. When one of these bits is displayed, the text indicates the appropriate user bit position rather than describing it as UNKNOWN.

The following display shows an example of the additions to the affected output display portions of a SHOW CPU/FULL command:

```
CPU 01 is in RUN state
Current Process: VMSADU          PID = 00000091
Serial Number:  AY24870406
Revision:       A200
VAX floating point operations supported.
IEEE floating point operations and data types supported.
PALCODE: Revision Code = 5.48
         PALcode Compatibility = 0
         Maximum Shared Processors = 2
         Memory Space:  Physical address = 00000000 00000000
                        Length = 0
         Scratch Space: Physical address = 00000000 00000000
                        Length = 0
Capabilities of this CPU:
        System:         QUORUM RUN
        User bitmask:   0000000F
Processes which can only execute on this CPU:
        VMSADU           PID = 00000091  Reason: Affinitized to this CPU ♦
```

2.2 High-Performance Sort/Merge Utility (Alpha Only)

Alpha

A new high-performance Sort/Merge utility takes advantage of the Alpha architecture to provide better performance for most sort and merge operations.

This section describes use of the high-performance Sort/Merge utility from the command line. Refer to Section 4.7 for information about the callable interface to the high-performance Sort/Merge utility by means of SOR$ routines.

Command Line Interface

The high-performance Sort/Merge utility uses the same command line interface as the SORT/MERGE utility. Many existing sort and merge operations can take advantage of the high-performance Sort/Merge utility without modification.

See the *OpenVMS User's Manual* for general information on using SORT /MERGE. See the Bookreader version of the *OpenVMS Version 7.0 Release Notes* for information related to problems and restrictions associated with this release of the high-performance Sort/Merge utility.

Selecting the High-Performance Sort/Merge Utility

Use the SORTSHR logical to select the high-performance Sort/Merge utility. Define SORTSHR to point to the high-performance sort executable in SYS$LIBRARY, as follows:

```
$ define sortshr sys$library:hypersort.exe
```

To return to the SORT/MERGE utility, deassign SORTSHR. The SORT/MERGE utility is the default if SORTSHR is not defined.

Using the High-Performance Sort/Merge Utility

The behavior of the high-performance Sort/Merge utility is the same as SORT /MERGE except as shown in Table 2–1.

Table 2–1 High-Performance Sort/Merge: Differences in Behavior

Feature	High-Performance Sort/Merge Behavior
Output file organization	Indexed sequential output file organization is not supported.[1] Do not specify the /INDEXED_SEQUENTIAL file organization qualifier. The /FORMAT, /RELATIVE, and /SEQUENTIAL output qualifiers are supported. The default is /SEQUENTIAL.
Key data types	The H-FLOATING and ZONED decimal data types are not supported. The size of a BINARY data type key must be 1, 2, 4, or 8 bytes. A 16-byte binary key is not supported.[1]
Collating sequences	The National Character Set (NCS) collating sequences are not supported.[1] Do not specify the name of an NCS collating sequence for the /COLLATING_SEQUENCE qualifier. The ASCII, EBCDIC, and MULTINATIONAL collating sequences are supported. The default is ASCII. You cannot define or modify your own collating sequence through the use of a specification file.[1]
Specification files	Specification files are not supported.[1] Do not use the /SPECIFICATION qualifier.
Internal sorting process	Only the record sort process is supported.[1] You can specify /PROCESS=RECORD or omit the /PROCESS qualifier. The TAG, ADDRESS, and INDEX values for the /PROCESS qualifier are not supported.
Statistical summary information	Statistical summary information is not supported.[1] Do not use the /STATISTICS qualifier.
Output file overlay	You cannot overlay or write the output file to an existing empty file. Do not specify the /OVERLAY qualifier.[1]

[1]Implementation of this feature is deferred to a future OpenVMS Alpha release.

If you attempt to use an unsupported qualifier or assign an unsupported value to a qualifier, the high-performance Sort/Merge utility generates an error.♦

2.3 Mail Utility

The following sections describe new features added to the OpenVMS Mail utility. These features include support for the following:

- Signature files
- /PAGE qualifier
- /NEXT qualifier

Note that all new commands and qualifiers are described in the DCL and Mail utility Help facilities as well.

2.3.1 Support for Signature Files

With this release of the OpenVMS operating system, you can now append text to the end of a mail message by including a **signature file**. An example of a signature file is a text file formatted as a business card, containing the user's company name, address, telephone, and Internet address.

You can set the Mail utility to include a signature file automatically (by default) with every mail message or you can selectively include a signature file only with particular mail messages. The following sections describe these and related operations.

2.3.1.1 Displaying Signature File Information

To determine if a signature file is currently set to be appended automatically to your mail messages, you can enter the SHOW SIGNATURE_FILE command or SHOW ALL command at the Mail prompt. For example:

```
MAIL> SHOW ALL
Your mail file directory is MAILD$:[MAILTU7].
Your current mail file is MAILD$:[MAILTU7]MAIL.MAI.
Your current mail folder is MAIL.
The wastebasket folder name is WASTEBASKET.
Mail file MAILD$:[MAILTU7]MAIL.MAI
        contains 0 deleted message bytes.

You have 0 new messages.

You have not set a forwarding address.
You have not set a personal name.
Your editor is EDT.
CC prompting is disabled.
Automatic copies to yourself are disabled.
Automatic deleted message purge is enabled.
Your default print queue is SYS$PRINT.
You have not specified a default print form.
Your default signature file is SECTION1.SIG.
```

If there is no signature file set, the last line in the display states the following:

```
        You have not specified a default signature file.
```

2.3.1.2 Including a Signature File by Default

To set Mail to automatically include a signature file with every message, use the SET SIGNATURE_FILE command. For example:

```
MAIL> SET SIGNATURE_FILE MY_ADDRESS.SIG
```

Note that if you do not specify a file type, the default is .SIG.

2.3.1.3 Disabling the Default Setting

If you have set Mail to include a signature file with your mail messages automatically, you can disable that default setting by using the SET NOSIGNATURE_FILE command. For example:

```
MAIL> SET NOSIGNATURE_FILE
```

If you want to *temporarily* disable the signature file setting while sending a particular mail message, you can use the /NOSIGNATURE_FILE qualifier with the following commands:

- ANSWER
- FORWARD
- MAIL
- REPLY
- SEND

If you want to temporarily disable the signature file setting from the DCL level, you can specify the /NOSIGNATURE_FILE qualifier with the DCL command MAIL.

2.3.1.4 Overriding the Default Setting

If you have set Mail to include a signature file with your mail message automatically but you want to append a *different* signature file, you can override the default setting by specifying the /SIGNATURE_FILE qualifier with the name of the signature file you want to include. For example:

```
MAIL> SEND/SIGNATURE_FILE=BUSINESS_CARD.SIG
```

You can specify the /SIGNATURE_FILE qualifier with the following Mail commands:

- ANSWER
- FORWARD
- MAIL
- REPLY
- SEND

You can also specify the /SIGNATURE_FILE with the DCL command MAIL.

2.3.1.5 Using Signature Files

When you use signature files, note the following:

- A mail message that includes a signature file requires more temporary disk space than a conventional message because temporary files are created during the operation. After the message is sent, those temporary files are deleted.
- If you specify a signature file that does not exist, the system displays an error message.
- If you do not specify a directory when you use the SET SIGNATURE_FILE command or /SIGNATURE_FILE qualifier, Mail searches for the file in your mail directory.
- If you do not specify a file when you use the /SIGNATURE_FILE qualifier, Mail uses the file specification in your user profile (if set previously with the SET SIGNATURE_FILE command). If there is no default signature file in the profile, Mail will send the message without one.

2.3.2 /PAGE Qualifier Now Available with Mail Commands

The /PAGE[=keyword] qualifier, which controls the display of information on the screen, is now valid with the following Mail commands:

- BACK
- CURRENT
- DIRECTORY
- FIRST
- LAST
- NEXT
- READ

You can use the following keywords with the /PAGE qualifier:

CLEAR_SCREEN	Clears the screen before each page is displayed.
SCROLL	Displays information one line at a time.
SAVE[=*n*]	Enables screen navigation of information, where *n* is the number of pages to store.

The /PAGE qualifier allows you to navigate through screens of information, storing up to 5 screens of up to 255 columns of information. When you use the /PAGE qualifier, you can use the following keys to navigate through the information:

Key Sequence	Description
Up arrow (↑), Ctrl/B	Scroll up one line.
Down arrow (↓)	Scroll down one line.
Left arrow (←)	Scroll left one column.
Right arrow (→)	Scroll right one column.
Find (E1)	Specify a string to find where the information is displayed.
Insert Here (E2)	Scroll right one half screen.
Remove (E3)	Scroll left one half screen.
Select (E4)	Toggle 80/132 column mode.
Prev Screen (E5)	Get the previous page of information.
Next Screen (E6), Return, Enter, Space	Get the next page of information.
F10, Ctrl/Z	Exit.
Do (F16)	Toggle the display to oldest/newest page.
Ctrl/W	Refresh the display.

Note that /NOPAGE is the default.

2.3.3 /NEXT Qualifier Valid with SEARCH Command

If you have a large number of messages, you can locate a particular message by using the SEARCH command to find a specified string. The SEARCH command selects and displays the first message that contains the specified string.

To continue searching for messages that contain the specified string, you can now specify the /NEXT qualifier with the SEARCH command. (You can also repeat the SEARCH command without specifying a parameter to find the next message containing the previously specified string).

2.4 Easy Internet Access

Since OpenVMS Version 6.2, OpenVMS has included Internet-ready features to make the Internet more easily accessible to OpenVMS users. OpenVMS Version 7.0 includes features that extend this access to the Internet.

Additionally, an OpenVMS Internet product suite is being launched as a separately orderable layered product to give VAX and Alpha customers the ability to launch an OpenVMS Web server using a wide assortment of leadership commercial Internet products and popular Internet freeware.

The product suite includes the following:

- Clients, including Netscape Navigator; Spyglass Enhanced Mosaic browser; lynx, a text-based WWW browser (freeware); mxrn, a Motif News reader (freeware); and a gopher client (freeware).
- Servers, including the Netscape Communications Server and Purveyor Server for high-performance Internet transactions; the Netscape Commerce Server for secure Internet transactions; Netscape Proxy Server; IUPOP3 mail server (freeware); a gopher server (freeware); and the ANU-News server (freeware).
- A firewall—Digital Firewall for OpenVMS (formerly known as Seal).
- Server applications, including a Web interface to VTX, Notes, and Monitor.
- Tools and Utilities ported to OpenVMS, including tcl, pearl, bison (a parser generator), grep, fgrep, gawk, flex, and sed.
- HTML tools for converting HLP files to HTML, checking HTML syntax, removing HTML markup, converting text to HTML, converting HTML to PostScript, searching and replacing in an HTML file, and converting SDML to HTML.

The OpenVMS Internet Product Suite Version 1.0, which requires OpenVMS Version 6.1 and DECwindows Motif Version 1.2-3 or later, will be available as a CD-only offering in early 1996. The CD purchase price will include the right to use all freeware. Third-party products as well as the Digital Firewall and Web Interface to VTX will be separately licensed, although each will have a 60-day trial offer PAK to let users test drive the software.

2.4.1 Internet Features Added to OpenVMS Systems

Since OpenVMS Version 6.2, Internet features have been added to make the Internet more easily accessible to OpenVMS users.

- The following Internet ready features were introduced in OpenVMS Version 6.2:
  - New DCL commands to allow TCP/IP users to use familiar DCL style syntax when performing TCP/IP activities such as ftp and telnet. (TCP/IP is the transport protocol used by the Internet.) These new DCL commands include COPY/FTP, COPY/RCP, DIRECTORY/FTP, SET HOST/RLOGIN, SET HOST/TELNET, and SET HOST/TN3270. See *TCP/IP Networking on OpenVMS Systems* for more information.
  - XTI front-end. XTI is a protocol independent network API library that allows you to develop protocol independent client/server applications. These applications using XTI can be run across any supported network transport which, on OpenVMS, includes DECnet and TCP/IP.
  - Support for clusters in a TCP/IP environment.
- The following Internet ready features are new in OpenVMS Version 7.0:
  - 4.4BSD Sockets. Sockets are the means by which services communicate between nodes over the Internet. Previously, OpenVMS implemented 4.3BSD Sockets. In OpenVMS Version 7.0, socket support has been extended to support 4.4BSD and XPG4 V2 routines for users writing Internet applications using TCP/IP services. See the *DEC C Run-Time Library Reference Manual for OpenVMS Systems* for more information.
  - New C RTL functions to make it easier to port UNIX applications that contain signals, timezones, string routines, random numbers, directory services, etc. See the *DEC C Run-Time Library Reference Manual for OpenVMS Systems* for more information.
  - OpenVMS Mail enhancements to:
    + Allow users to append a signature file to a mail message. An example of a signature file is a text file formatted as if it were a business card that contains the user's company name, address, telephone number, and Internet address. See the section titled "Support for Signature Files" in the *OpenVMS Version 7.0 New Features Manual* for more information.
    + Allow users to more easily use the SMTP protocol—which is part of the TCP/IP transport protocol used by the Internet—in addition to DECnet. See the *OpenVMS Version 7.0 Release Notes* for more information about specifying mail transports.
  - Monitor enhancements to support other transports—for example, TCP/IP—in addition to DECnet. You can now monitor any system within a cluster whether that cluster is running TCP/IP or DECnet. See the *OpenVMS Version 7.0 Release Notes* for more information.

  - Integrated network support to provide a common DCL interface to display information about starting, stopping, and displaying network services information about DECnet, DECnet/OSI, and TCP/IP Services for OpenVMS. The new DCL commands to obtain this information are SHOW NETWORK, START/NETWORK, STOP/NETWORK, and SET NETWORK. These new commands are described in *OpenVMS DCL Dictionary: N–Z.*

- Future releases of OpenVMS will also include support for these additional Internet-ready features:
  - DCL pipes that, using UNIX style and syntax, allow you to redirect the output of one DCL command to another.
  - PPP, or Point-to-Point Protocol, which provides dial-in and dial-out support for asynchronous serial lines as well as a utility to configure, start, and stop PPP on a line-by-line basis.
  - Kerberos Authentication service, which provides users with a trusted third-party authentication system. The need for such services helps to minimize the vulnerability of using unprotected passwords over an insecure network.

3
System Management Features

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

3.1 New and Changed System Parameters

The following sections describe the most important new OpenVMS system parameters in these two categories:

- System parameters, which you can modify
- SPECIAL parameters, which you should modify only if Digital recommends it

Descriptions of additional new system parameters and changes to existing parameters are in Help and will appear in the next printed version of the *OpenVMS System Management Utilities Reference Manual*.

3.1.1 System Parameters

Following is an alphabetical list of important new system parameters. Use the descriptions as guidelines for deciding whether or not to modify these parameters.

3.1.1.1 CWCREPRC_ENABLE

The CWCREPRC_ENABLE system parameter controls whether an unprivileged user can create a process on another VMScluster node. The default value of 1 allows an unprivileged user to create a detached process with the same UIC on another node. A value of 0 requires that a user have DETACH or CMKRNL privilege to create a process on another node.

3.1.1.2 DBGTK_SCRATCH (Alpha Only)

Alpha

(This parameter was renamed from RMTDBG_SCRATCH_PAGES.)

On Alpha systems, the DBGTK_SCRATCH system parameter specifies how many pages of memory are allocated for the remote debugger. This memory is allocated only if remote debugging is enabled with the 8000 boot flag. Normally, the default value is adequate, but if the remote debugger issues an error message, you should increase this value. See the *OpenVMS Alpha Device Support Manual* for more information.♦

3.1.1.3 IO_PREFER_CPUS (Alpha Only)

Alpha

On Alpha systems, the IO_PREFER_CPUS system parameter excludes processors from being used as Preferred CPUs for Fast Path I/O. IO_PREFER_CPUS is a 32-bit mask; if the value of a bit in the mask is 1, the processor with the corresponding CPU ID is available to be used as a Preferred CPU.

The default value of IO_PREFER_CPUS, -1, allows all available CPUs to become Preferred CPUs. The parameter is used only if Fast Path is enabled. See the FAST_PATH special parameter. ♦

3.1.1.4 MAXBOBMEM (Alpha Only)

Alpha

On Alpha systems, MAXBOBMEM determines the maximum number of pagelets that can be made into buffer objects system-wide. MAXBOBMEM is a DYNAMIC parameter.♦

3.1.1.5 MULTITHREAD (Alpha Only)

Alpha

On Alpha systems, MULTITHREAD controls the availability of kernel threads functions. Specify one of the following values:

Value	Description
0	Both Thread Manager upcalls and the creation of multiple kernel threads are disabled.
1	Thread Manager upcalls are enabled; the creation of multiple kernel threads is disabled.
2-16	Both Thread Manager upcalls and the creation of multiple kernel threads are enabled. The number specified represents the maximum number of kernel threads that can be created for a single process.♦

See Section 4.8 for more information about kernel threads.

3.1.2 SPECIAL Parameters

These new parameters are subject to change at any time and should be modified only if recommended by Digital.

3.1.2.1 FAST_PATH (Alpha Only)

Alpha

On Alpha systems, FAST_PATH enables (1) or disables (0) Fast Path I/O on a system. See the IO_PREFER_CPUS system parameter. In OpenVMS Alpha Version 7.0, FAST_PATH only supports disk I/O for the CIXCD port.

For more information about how to use FAST_PATH, see Chapter 5. ♦

3.1.3 Changed System Parameters

The following section describes changes to an existing system parameter.

3.1.3.1 ACP_DATACHECK Has Three New Levels

The ACP_DATACHECK system parameter has three new levels for checking reads and writes of directory blocks. The new default checks reads only.

You can select a level by setting bits 5 and 6 as follows:

To Check That...	Select This Level...	By Setting Bit 6 to...	And Bit 5 to...
The block is a valid directory block (reads only).	0 (Default)	0	0
The block is a valid directory block (reads and writes).	1	0	1
The block is a valid directory block, and that it contains valid entries (reads and writes).	2	1	0

To Check That...	Select This Level...	By Setting Bit 6 to...	And Bit 5 to...
The block is a valid directory block, and that it contains valid entries in correct alphanumeric order (reads and writes).	3	1	1

The ACP_DATACHECK system parameter is dynamic; checking starts on the first read or write after you have selected a level.

When you set the SYSTEM_CHECK system parameter to 1, you enable level 3 checking.

Write errors result in BUGCHECK and crash your system, read errors exit with error status SS$_BADDIRECTORY.

DUMPSTYLE (A,D)

DUMPSTYLE specifies the method of writing system dumps.

DUMPSTYLE is a 32-bit mask, with the following bits defined. Each bit can be set independently. The value of the SYSGEN parameter is the sum of the values of the bits that have been set. Remaining or undefined values are reserved to Digital.

Bit	Value	Description
0	0	0 = Full dump (SYSGEN default). The entire contents of physical memory will be written to the dump file.
		1 = Selective dump. The contents of memory will be written to the dump file selectively to maximize the usefulness of the dump file while conserving disk space.
1	2	0 = Minimal console output.
		1 = Full console output (includes stack dump, register contents, and so on).
2 (VAX only)[1]	4	0 = Dump to system disk.
		1 = Dump to alternate disk.
3 (Alpha only)[2]	8	0 = Do not compress.
		1 = Compress. (See note below.)

[1] Alpha systems do not support dumping to an alternate disk.

[2] VAX systems do not support dump compression.

Note

Alpha

On Alpha systems, system managers can save space on the system disk and, in the event of a crash, save time recording the system memory, by using the OpenVMS Alpha dump compression feature. Unless the system manager overrides the default AUTOGEN calculations (by setting DUMPSTYLE in MODPARAMS.DAT), AUTOGEN uses the following algorithm:

- On a system with less than 128 MB of memory, the system sets the DUMPSTYLE to 1 (a raw selective dump) and sizes the dump file appropriately.
- On a system with 128 MB of memory or greater, the system sets the DUMPSTYLE to 9 (a compressed selective dump), and creates the dump file at two-thirds the value of the corresponding raw dump.♦

Examples

VAX

On VAX systems, the value of 4 directs the system to send a full dump, with minimal console output, to the alternate disk.♦

Alpha

On Alpha systems, the value of 9 directs the system to compress a selective dump.♦

AUTOGEN evaluates mathematical expressions; for example, the expression DUMPSTYLE = 4 + 2 in MODPARAMS.DAT is evaluated as 6.

3.2 LAT New Features

With Version 7.0 of the OpenVMS operating system, the LAT software has been upgraded to LAT protocol Version 5.3. Note that because Version 5.3 of the LAT protocol is compatible with Versions 5.1 and 5.2 of the LAT protocol, you can still connect to OpenVMS systems that are running those older versions of the LAT protocol (5.1 or 5.2).

The following sections describe new features associated with Version 5.3 of the LAT protocol. These features include the following:

- Displaying faster terminal speeds
- Using multiple LAN adapters
- Managing large buffers
- Controlling service announcements
- Using a user-written LAT rating algorithm
- Specifying a new SET HOST/LAT qualifier (/FRAME)

3.2.1 Displaying Speeds Greater than 57,600 Kbps

Data B slot modifications have been made so that terminal input and output speed can be reported (correctly) when the terminal speed is in excess of a 16-bit value. If both sides of the LAT connection (including the terminal server) are running Version 5.3 of the LAT software, you can use the DCL command SHOW TERMINAL to display speed values greater than 57,600 kbps.

3.2.2 Using Multiple LAN Adapters

With this release, you can now use multiple LAN addresses for one LAT node. This enables a system manager to configure a system with multiple LAN adapters connected to the same logical LAN. The LAT software can run over each adapter simultaneously. The following sections provide more detail about this new feature.

Note

Nodes running versions of LAT software prior to Version 5.3 of the LAT protocol (which was included in the OpenVMS operating system beginning with Version 7.0) may exhibit some differences in behavior. Therefore, if your configuration includes earlier versions of the LAT software, such as Version 5.1 or Version 5.2, note the differences and considerations discussed in this section.

3.2.2.1 Multiple LAN Addresses

It is now possible for a LAT Version 5.3 master node (which *accesses* services) and a LAT Version 5.3 node that *provides* services to both use multiple LAN addresses. (Nodes running LAT Version 5.2 and Version 5.1 are still restricted to one LAN address per logical network.)

This new feature allows the LAT software to maintain connections. For example, when a virtual circuit chooses a primary path and uses it for all LAT message transmissions, the LAT software now has the ability to continue communications through another adapter or logical path if that original path becomes blocked.

3.2.2.2 Supported Configurations

Although it is possible to run LAT over multiple LAN adapters, it is still not possible to route LAT from one logical LAN to another. The following four figures are examples of supported LAT configurations for nodes running Version 5.3 of the LAT protocol (including nodes running Version 5.2 and 5.1 as well).

Figure 3–1 Multiple Address LAT Configuration: One LAN with Mixed Version LAT Nodes

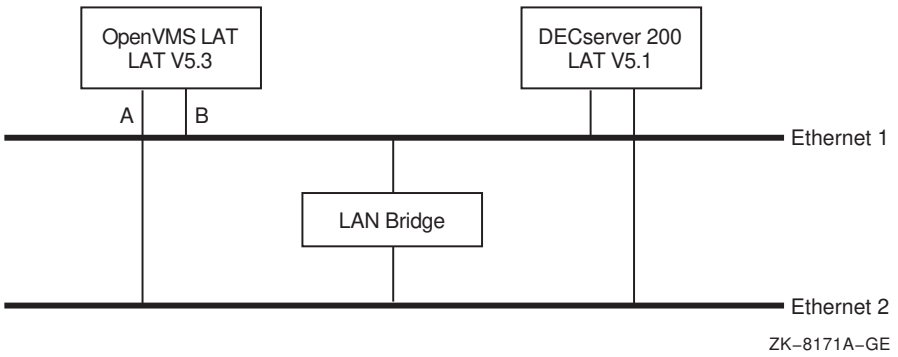
This widely used configuration has an OpenVMS system running LAT Version 5.3 over two Ethernet adapters (labeled A and B in the diagram) connected to the same physical LAN as a DECserver 200.

When a LAT connection is started between the DECserver 200 and the OpenVMS system, LAT Version 5.3 determines that it is possible to use both adapters A and B for the LAT virtual circuit. One of the adapters will be chosen as the primary communications path while the other will be present in the unlikely event that the primary path fails.

For example, if a user connects to the OpenVMS system from the DECserver 200, the OpenVMS system determines that there are two paths but chooses adapter B as the primary communications path. If the user runs a program that generates a large amount of output from the OpenVMS system and adapter B fails in some manner during the output, the LAT software will attempt to continue communications from the OpenVMS system to the DECserver through adapter A.

Figure 3–2 shows two LANs bridged together. However, this configuration will have the same characteristics as the configuration shown in Figure 3–1.

Figure 3–2 Multiple Address LAT Configuration: Two LANs with Mixed Version LAT Nodes

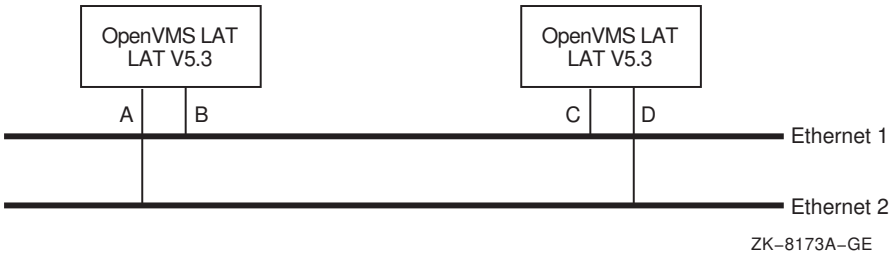


Note

It is possible for Ethernet 2 in Figure 3–2 to be an FDDI network. The LAT software regards each adapter as a network path with equal cost, point-to-point communications and does not treat FDDI controllers any differently. However, for large buffer support, see Section 3.2.4 for more details.

In the configuration shown in Figure 3–3, any virtual circuit created between the two OpenVMS systems will have two paths: through controllers B and C or A and D. If one path fails, the virtual circuit will continue over the other path. If both paths fail, the virtual circuit will eventually time out.

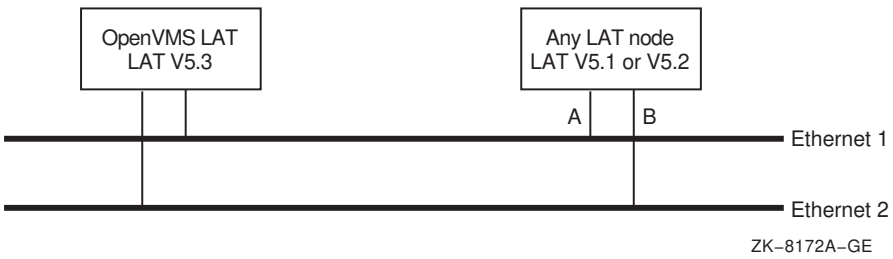
Figure 3–3 Multiple Address LAT Configuration: Two LANs with Version 5.3 LAT Nodes



3.2.2.3 Unsupported Configuration

When configuring a network to use an OpenVMS system running LAT Version 5.3, *avoid* the configuration shown in Figure 3–4.

Figure 3–4 Unsupported Multiple Address LAT Configuration



Any configuration similar to this diagram will result in unpredictable results and may not function. In a network environment, LAT Version 5.1 and Version 5.2 nodes can have only a single logical LAN address. The configuration in Figure 3–4 violates this rule. The configuration shown in Figure 3–3 is a valid alternative.

3.2.2.4 Creating Logical LAT Links

The LAT software regards all paths as equal cost, point-to-point communication. The LAT software can support a maximum of eight LAN adapters simultaneously (and it is possible to connect all controllers to the same logical LAN). To get the maximum coverage over possible path failures, each logical link should be created prior to setting the LAT node state to ON in SYS$MANAGER:LAT$SYSTARTUP.COM.

For example, if a system has one Ethernet adapter (device ESA0) with two FDDI adapters (FCA0 and FCB0) and the system manager chooses to run LAT over all adapters, the LAT$SYSTARTUP.COM file would contain the following commands:

```
$!
$! Create each logical LAT link with a unique name and
$! unique LAN address (forced with /NODECNET).
$!
$ LCP CREATE LINK ETHERNET /DEVICE=ESA0 /NODECNET
$ LCP CREATE LINK FDDI_1 /DEVICE=FCA0 /NODECNET
$ LCP CREATE LINK FDDI_2 /DEVICE=FCB0 /NODECNET
$!
$! Turn on the LAT protocol.
$!
$ LCP SET NODE /STATE=ON
```

Caution

If the LATCP command SET NODE /STATE=ON is entered before the link is created, a random or default LAT$LINK will be created on one of the LAN adapters. There is no way to predict which LAN adapter will be chosen (it is dependent on the system configuration). Therefore, all logical LAT links should be created before LAT is started.

Be sure each logical link is created with the /NODECNET qualifier. It will prevent the possibility of link creation failure if multiple adapters attempt to use the DECnet style address. Having more than one LAN adapter connected to the same logical LAN with the same address violates LAN conventions and will cause problems with LAT and other protocols.

It is possible to create logical LAT datalinks after the LAT protocol has been started. Any existing virtual circuit will attempt to find any new paths through the newly created logical datalink when it is ready for use. However, Digital does not recommend that you create links at this point because during the time it takes existing virtual circuits to discover new paths through this newly created datalink, the virtual circuit may fail before the new paths are discovered.

3.2.2.5 Path Discovery

The OpenVMS LAT software uses a combination of the directory service and solicitation to obtain paths for each virtual circuit. Digital recommends that a system with multiple LAN adapters be configured to maintain a LAT service and node database. This expedites path discovery at virtual circuit startup. Recommendations for doing this include:

- Enabling outgoing LAT connections
- Using the same group code mask for User Groups and Service Groups

An OpenVMS system running as a LAT node that provides services only (outgoing connections disabled and no service or node database) is still capable of running with multiple paths for each virtual circuit. These paths must be discovered through the LAT solicitation process and will take longer (leaving the possibility for virtual circuit failure to occur before all paths have been discovered).

3.2.3 Modifying LAT Parameters

In the unlikely event of a path failure, it will take the OpenVMS LAT software time (which will vary depending on the number of adapters to which the remote node has access) to locate another working path. Therefore, Digital strongly recommends that you modify the following LAT parameters on potential LAT master nodes:

- Retransmit limit - default value is 8. Set to the maximum number of LAN adapters times 8. For example, if an OpenVMS system on the LAN has 3 adapters, each LAT master node should have their retransmit limit set to 3 * 8 = 24.
- Keepalive timer - default value is 20 seconds. While the default value may be sufficient in most circumstances, it may be necessary to increase this to 30 or 40 seconds.

Although it is possible to keep virtual circuits running through multiple adapters to LAT Version 5.1 or LAT Version 5.2 master nodes, there is still a possibility that the connections to these nodes may fail.

LAT Version 5.2 and LAT Version 5.1 master nodes do not have the ability to recognize multiple paths to LAT nodes that provide services. They can only communicate with such nodes through one remote address at a time. Therefore, if a LAN path failure occurs when a LAT master node running LAT Version 5.1 or Version 5.2 attempts to connect to a remote LAT Version 5.3 node providing services, the LAT Version 5.3 node might not discover this failure in time and the LAT master node may time out the connection. You can partially solve this problem by increasing the retransmit limit to as high a setting as possible.

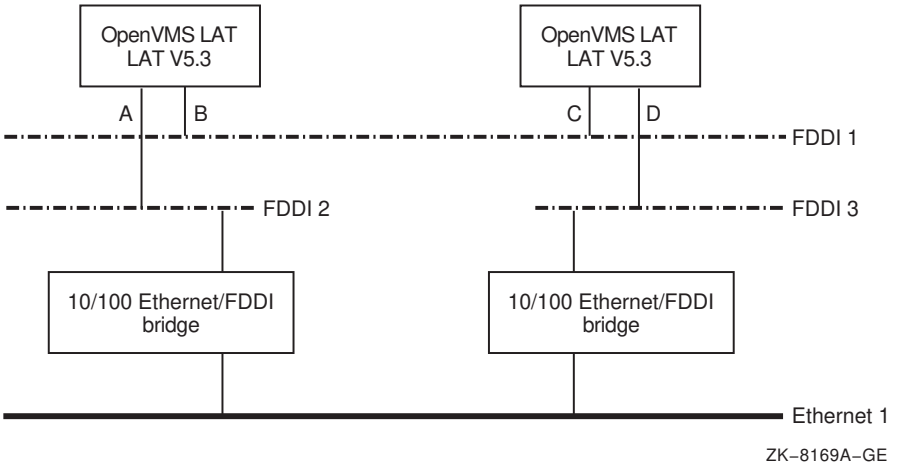
In addition, if a LAT Version 5.3 node providing services views the virtual circuit as completely idle during the primary path failure, no attempt will be made to use any of the alternate paths (because of the previously described LAT Version 5.2 and 5.1 limitation). Therefore, although multiple LAN adapters will work with older LAT implementations, you might still need to upgrade to the OpenVMS Version 7.0 operating system (which includes the LAT Version 5.3 protocol) to correct this type of problem. (Note that this type of problem affects only those connections that are idle. An example of where this situation could arise is in an office environment if all users were to leave their systems at the same time, either at lunchtime or at the end of the workday.)

3.2.4 Managing Large Buffers

The OpenVMS LAT software will attempt to use *large* buffers over any virtual circuit that comes in over an FDDI controller. This feature can cause problems if an alternate virtual circuit path must go through an Ethernet. Figure 3–5 is an example of the configuration that can cause problems.

In this diagram, it is possible for the two OpenVMS systems to communicate using large packets through the path described by controllers B and C. *Large* packets are those that may exceed 1500 bytes of data (the maximum Ethernet message can contain 1500 bytes of data). If the path described by controllers B and C were to fail, it would not be possible for communication to continue through the path described by A and D.

Figure 3–5 LAT FDDI Ring and Large Buffers



The path described by controllers A and D pass through an Ethernet LAN segment. The messages that are routed through the 10/100 bridges cannot be larger than the maximum Ethernet message. Problems can occur because the OpenVMS LAT software cannot always detect this kind of configuration.

There are two ways to prevent problems with the previously described configuration. The first and easiest option is to create a logical LAT link using an Ethernet adapter (if either system has an Ethernet LAN adapter). This will force the message size negotiation to be no larger than the maximum sized Ethernet message.

If neither system has an Ethernet controller (thus making the first option not possible), the second option is to override the use of large buffer support (which is enabled by default) by using the new LATCP command qualifier, /[NO]LARGE_BUFFER. For example:

```
$ MCR LATCP SET NODE/NOLARGE_BUFFER
```

Digital recommends that you use the SET NODE/NOLARGE_BUFFER command after all logical LAT links have been created and before the LAT node has been turned on. For example, note the order of the commands in LAT$SYSTARTUP.COM:

```
$!
$! Create each logical LAT link with a unique name and
$! unique LAN address (forced with /NODECNET).
$!
$ LCP CREATE LINK FDDI_1 /DEVICE=FCA0 /NODECNET
$ LCP CREATE LINK FDDI_2 /DEVICE=FCB0 /NODECNET
$!
$! Don't use large buffer support (force packet
$! sizes to be no larger than what Ethernet can
$! support).
$!
$ LCP SET NODE /NOLARGE_BUFFER
$!
$! Turn on the LAT protocol.
$!
$ LCP SET NODE /STATE=ON
```

3.2.5 Controlling Service Announcements

With this release, you can now use the /[NO]ANNOUNCEMENTS qualifier with the LATCP command SET NODE to control whether your OpenVMS system multicasts information to the network. For example:

```
$ LCP SET NODE /NOANNOUNCEMENTS ! Disabled service announcements
```

Note that, because remote nodes must rely on the LAT service responder feature in the LAT protocol Version 5.2 (or higher) to connect to the local node, Digital recommends that you use this qualifier only in a networking environment where newer model terminal servers and hosts are present (all LAT hosts, terminal servers, and PCs are running at least Version 5.2 of the LAT protocol). Otherwise, systems running versions of the LAT protocol prior to Version 5.2 (for example, DECserver 100, 200, and 500 systems) will be unable to connect to any of the systems that have LAT service announcements disabled.)

3.2.6 Support for User-Written LAT Rating Algorithm

The LAT rating algorithm is no longer part of LTDRIVER. Instead, a separate loadable image (LAT$RATING.EXE) containing the logic for the LAT rating algorithm is provided in SYS$LOADABLE_IMAGES. This image is loaded by LAT$CONFIG.COM.

The sources to produce LAT$RATING.EXE and an example command procedure for compiling and linking those sources are provided in SYS$EXAMPLES. You need either a VAX C compiler or a DEC C compiler to build LAT$RATING. The sources in SYS$EXAMPLES contain code for the default LAT rating algorithm.

If you want to modify the LAT rating algorithm to suit a specific configuration, be sure you examine the source code to understand the relationship between LAT$RATING and LTDRIVER before you make any modifications.

Caution

To prevent a system crash, only a system programmer experienced with OpenVMS device drivers should modify the LAT rating image. Inspect the resulting C compiler listing to ensure that the generated instructions are legal for an OpenVMS device driver.

Do **not** call any system services from the LAT rating image. (See the device driver documentation for additional guidelines.)

Table 3–1 describes the files provided in SYS$EXAMPLES for the LAT$RATING image contained in the OpenVMS operating system.

Table 3–1 LAT$RATING Sources

File Name	Description
LAT$RATING_BUILD.COM	Command procedure used to build the LAT$RATING image. The image itself must be copied into SYS$LOADABLE_IMAGES.
LAT$RATING_CALC.C	Source file that houses the LAT rating and load average calculations.
LAT$RATING_DPT.MAR	Source file that contains the necessary driver tables and special routine entry points that cannot be written in C for LAT$RATING.EXE.

3.2.7 New SET HOST/LAT Qualifier

The new /FRAME=n qualifier allows a user making a LAT connection to a remote system to specify the number of data bits that the terminal driver expects for every character that is input or output. The value of *n* can be from 5 to 8. The default value depends on the settings for the terminal established by the /PARITY and /EIGHT_BIT qualifiers. The following example specifies a character frame size of 7 bits per character:

```
$ SET HOST/LAT /FRAME=7 DIAL_OUT_SVC
```

3.2.8 New LAT Item Codes

There are also new programming item codes, described in detail in Section 4.11.

3.3 Networking Support

Beginning with OpenVMS Version 7.0, the following networking products are integrated into the installation and upgrade procedures for OpenVMS VAX and OpenVMS Alpha:

- DECnet Phase IV
- DECnet/OSI
- Digital TCP/IP Services for OpenVMS

For OpenVMS VAX Version 7.0 and OpenVMS Alpha Version 7.0, DECnet Phase IV is the default networking protocol.

For more information about installing these products with the operating system, see the *OpenVMS Alpha Version 7.0 Upgrade and Installation Manual* and the *OpenVMS VAX Version 7.0 Upgrade and Installation Manual*.

3.4 New Network Commands

It is possible to run more than one network service on an OpenVMS system. A revised SHOW NETWORK command and three new commands allow the user to define and display data about the various network services available on a particular machine, and to start or stop any of those services.

The three new network commands are:

- SET NETWORK
- START NETWORK
- STOP NETWORK

3.5 New Logical Names for OPCOM (Operator Communication Manager)

The following logical names have been added to the SYS$MANAGER:SYLOGICALS.COM command procedure. These logical names allow you to control the flow of OPCOM messages.

- OPC$ALLOW_INBOUND—Allows OPCOM traffic that is inbound to the node to be turned on or off. By default, this logical name is set to TRUE. If this logical name is set to FALSE, all OPCOM messages from other nodes in the cluster will not be received by the node.

- OPC$ALLOW_OUTBOUND—Allows OPCOM traffic that is outbound from the node to be turned on or off. By default, this logical name is set to TRUE. If this logical name is set to FALSE, all OPCOM messages from the node will not be sent to other nodes in the cluster.

Caution

Setting these logical names to FALSE severs all OPCOM traffic in the specified direction. All OPCOM messages, as well as any returned status messages that might be expected, will not be delivered.

3.6 Setting Correct Time Zone Information on Your System

Beginning with OpenVMS Version 7.0, the DEC C RTL implements its default date/time support for programs compiled with DEC C Version 5.2 using a model based on **Coordinated Universal Time** (UTC), an international standard for measuring time of day.

Caution

Even if you do not use the DEC C RTL directly, you *must* set correct time zone information on your system because other system utilities written in the DEC C language might require it.

To set the correct time zone information on your system, use the UTC$TIME_SETUP.COM command procedure to do the following:

1. Set the local time zone for your system.
2. Set the correct **time differential factor** (TDF) for your system.

Using UTC allows the DEC C RTL to implement ANSI C/POSIX functionality. In addition, the UTC model makes the DEC C RTL compatible with the Digital UNIX and POSIX RTL time functions. With a UTC-based system, users can do the following:

- Compute the time in any time zone
- Correctly compute past and future times, taking daylight saving time into account
- Use the ANSI C gmtime routine and make use of the tm-isdst field of the tm structure

The following sections explain these concepts and tasks.

Concept or Task	Section
Understanding time-setting concepts	Section 3.6.1
• Coordinated Universal Time (UTC)	
• Time differential factor (TDF)	
• Daylight saving time and standard time	
• Time zones	
Determining your system's TDF	Section 3.6.2
Using UTC$TIME_SETUP.COM	Section 3.6.3
• Setting the time zone on your system	
• Setting the TDF on your system	
Adjusting for daylight saving time and standard time	Section 3.6.4
Setting time in a VMScluster environment	Section 3.6.5

3.6.1 Understanding Time-Setting Concepts

Understanding some time concepts will help you see the importance of setting the correct time zone and TDF on your system.

3.6.1.1 Coordinated Universal Time

Coordinated Universal Time (UTC) is similar in most respects to Greenwich Mean Time (GMT). Under the UTC time standard, zero hours occurs when the Greenwich Meridian is at midnight. Unlike local time, which can go backward and forward depending on daylight saving time, UTC always increases.

Local times can be up to 12 hours behind Greenwich Mean Time or 13 hours ahead of it.

Because UTC is independent of time zones, you can use UTC around the world; for example, it is 2:00 UTC at the same moment in Paris as well as in Tokyo. You can examine data that is time-stamped with UTC values in Paris and Tokyo without complicated conversions to deal with local time zones.

3.6.1.2 Time Differential Factor

One of the steps in setting the correct time on your system is to calculate a time differential factor (TDF) for your time zone.

The TDF associates each local time zone with UTC; it is the difference between your local system time and UTC. The TDF changes each time your local system time undergoes a time zone change; the UTC, on the other hand, does not change.

The TDF value is expressed in signed (+ or -) *hh:mm* format. The Americas have negative TDFs, while Europe, Africa, Asia, and Australia have positive TDFs.

Section 3.6.2 explains how to select the correct TDF for your time zone.

3.6.1.3 Daylight Saving Time and Standard Time

Typically, you make seasonal changes to the local system time (for example, for daylight saving time and standard time). You usually adjust the local time one hour forward or backward.

You also need to adjust the TDF to compensate for the new local system time. You adjust the TDF in the same direction as the local time; that is, when you add an hour to the local time, you also add an hour to the TDF.

3.6.1.4 Time Zones

Time zones are names for geographical areas that share the same TDF; they also share the same rule or rules for seasonal changes between standard time and daylight saving time.

3.6.2 Determining Your System's Time Differential Factor

You can use the map in Figure 3–6 to determine the TDF for your time zone. If you prefer, you can use the tables in Appendix B in the *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems* to determine the standard or daylight saving time TDF for your time zone. The procedure described in Section 3.6.3 shows default TDF values for various time zones.

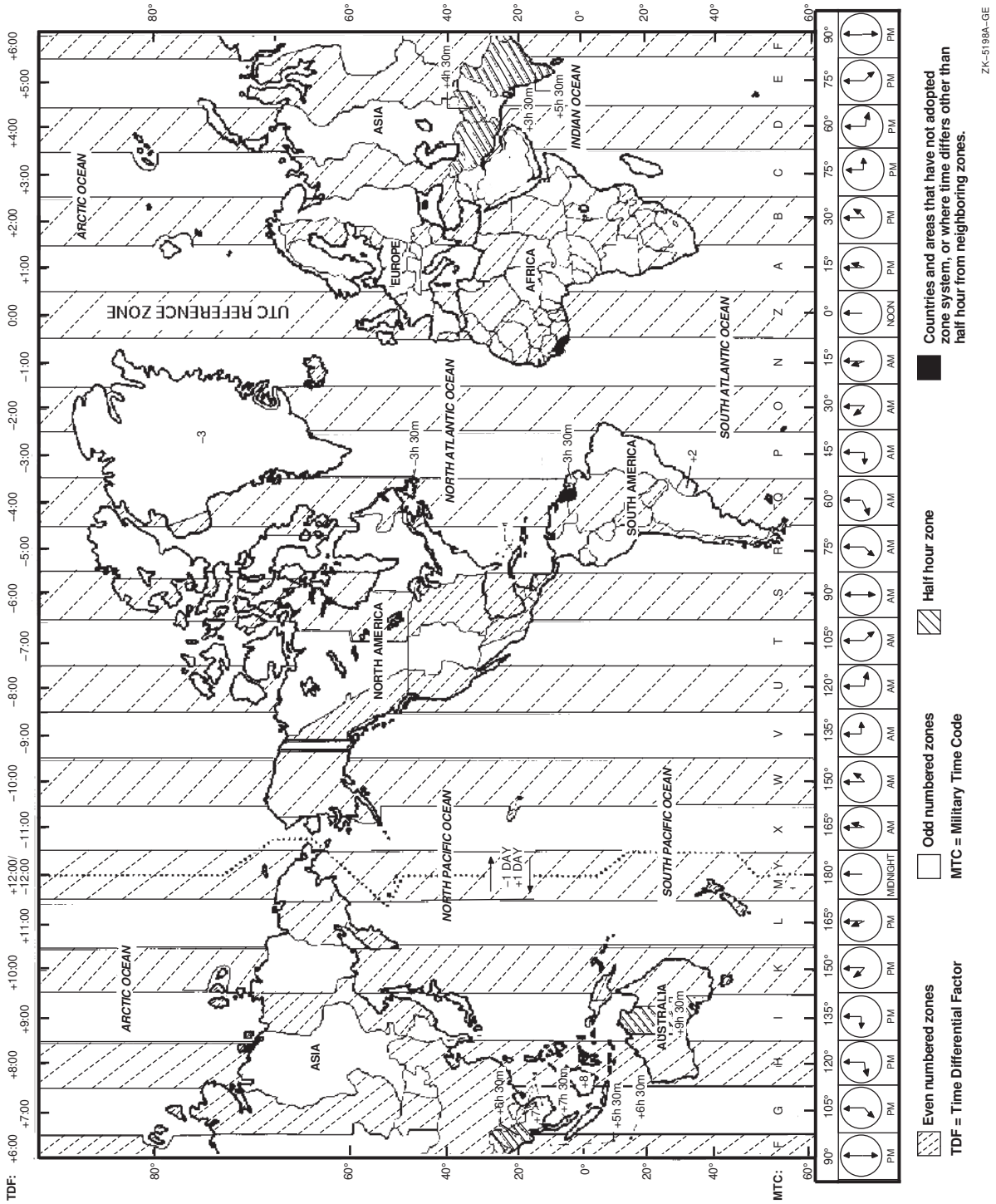
To use the map to determine the TDF of your time zone, follow these steps:

1. Find your location on the map and notice the time zone band at that location.
2. Follow the time zone band to the top of the map, and note the TDF that corresponds to your time zone. For example, the TDF for California is -8; the TDF for Italy is +1.

Some time zones do not have full-hour TDFs. In these cases, find the specific value on the map itself. For example, if you live in Adelaide, Australia, your TDF is +9:30.

If your time zone has daylight saving time, your TDF for daylight saving time is typically +1:00 from the standard time. For example, if your standard time TDF is +2:00, your daylight saving time TDF is +3:00; if your standard time TDF is -7:00, your daylight saving time TDF is -6:00.

Figure 3–6 Time Differential Factor Map



3.6.3 Using UTC$TIME_SETUP.COM

You can use this command procedure to set your local time zone or your TDF, or both. You can also use the procedure to modify your system's local time to adjust for daylight saving or standard time. Finally, you can use it to display the local time and TDF for your system.

If you set the time zone and the TDF on one node in a cluster, the values you set take effect on other nodes in the cluster when those nodes are rebooted.

For your convenience, the instructions for using the command procedure have been split into two sections:

- Setting the time zone
- Setting the TDF

Beginning the Procedure

To use SYS$MANAGER:UTC$TIME_SETUP.COM, follow these steps:

1. Log in to the SYSTEM account, or enter the following command to enable LOG_IO and OPER privileges:

   ```
   $ SET PROCESS/PRIVILEGE=(LOG_IO,OPER)
   ```

2. Enter the following command to start UTC$TIME_SETUP.COM:

   ```
   $ @SYS$MANAGER:UTC$TIME_SETUP.COM

   %UTC-I-UPDTIME, updating Time Zone information in SYS$COMMON:[SYSEXE]
   ```

3. Press Return to accept the default of BOTH, or enter one of the other choices in answer to the following question:

   ```
   Configure which time parameter (TIMEZONE/TDF/BOTH/NONE)? [BOTH]
   ```

   This question is asked only if you previously configured both the time zone and TDF. If either one has not been set, the system defaults to BOTH and the question is not asked.

   Note

   Digital recommends that you set BOTH the time zone and the TDF. If you set the TDF *without* setting the time zone, the procedure cannot provide default TDF values.

If you answer BOTH or TIMEZONE to the time parameter question in the command procedure, continue with the next section. If you answer TDF to the question, skip to Section 3.6.3.2.

3.6.3.1 Setting the Time Zone on Your System

The local time zone is the location you want to consider your default local time zone. Usually this local time zone is the same as the local time zone in which your system is located.

You set the local time zone by making choices in a command procedure.

The system first displays the following information:

```
Configuring the Local Time Zone

TIME ZONE SPECIFICATION -- Main Time Zone Menu

  1) Australia       11) GMT            21) Mexico         31) Turkey
  2) Brazil          12) Greenwich      22) NZ             32) UCT
  3) CET             13) Hong Kong      23) NZ-CHAT        33) US
  4) Canada          14) Iceland        24) Navajo         34) UTC
  5) Chile           15) Iran           25) PRC            35) Universal
  6) Cuba            16) Israel         26) Poland         36) W-SU
  7) EET             17) Jamaica        27) ROC            37) WET
  8) Egypt           18) Japan          28) ROK            38) Zulu
  9) Factory         19) Libya          29) Singapore
 10) GB-Eire         20) MET            30) SystemV

  0) None of the above
```

Table 3–2 lists and describes the acronyms that appear in the Main Time Zone Menu.

Table 3–2 Time Zone Acronyms

Time Zone Acronym	Description
CET	Central European Time
EET	Eastern European Time
Factory	Specifies no time zone
GB-Eire	Great Britain/Ireland
GMT	Greenwich Mean Time
NZ	New Zealand
NZ-CHAT	New Zealand, Chatham Islands
MET	Middle European Time
PRC	Peoples Republic of China
ROC	Republic of China
ROK	Republic of Korea
SystemV	Specific to System V operating system
UCT	Coordinated Universal Time
US	United States
UTC	Coordinated Universal Time
Universal	Coordinated Universal Time
W-SU	Middle European Time
WET	Western European Time

To select a time zone, follow these steps:

1. Enter a number after the following question; for example, 33, for the United States:

   ```
   Select the number above that best describes your location: 33
   ```

   If you enter 0, the system defaults to GMT.

2. If you enter a country that has more than one time zone, the system displays a message and asks for a confirmation like the following (if not, skip to the explanation following step 4):

   ```
   You selected US as your time zone.
   Is this correct? (Yes/No) [YES]: Return
   ```

3. The system displays areas with different time zones and asks you to select your area. Enter a number, for example, 6:

   ```
   US Time Zone Menu

      1) Alaska           4) Central          7) Hawaii          10) Mountain
      2) Aleutian         5) East-Indiana     8) Indiana-Starke  11) Pacific
      3) Arizona          6) Eastern          9) Michigan        12) Samoa

      0) None of the above

   Select the number above that best describes your location: 6
   ```

4. Confirm the displayed information, or enter another number after the following; for example:

   ```
   You selected US/Eastern as your time zone.
   Is this correct? (Yes/No) [YES]: Return
   ```

   When you confirm the last statement, the system redefines the following system logical names:

   - `sys$localtime`
   - `sys$posixrules`

   The DEC C RTL uses these logical names to compute the time zone rules for your applications. The system also saves this information and uses it to reset sys$localtime and sys$posixrules when you reboot.

   The system then displays the TDFs for standard and daylight saving time that correspond to the time zone you have selected:

   ```
    Default Time Differential Factor for standard time is -5:00.
   *Default Time Differential Factor for daylight saving time is -4:00.
   ```

   * The system displays daylight saving time only for time zones that use daylight saving time.

3.6.3.2 Setting the TDF on Your System

If you answer TDF or BOTH to the time parameter question at the beginning of the command procedure, the system displays prompts for the TDF on your system.

To set the TDF, answer the following questions:

1. Select option 2 to display the TDF the system has calculated for you:

```
    Configuring the Time Differential Factor (TDF)

    Enter ? anytime for help

    [0]     Exit
    [1]     Set the Time Differential Factor
    [2]     Display the Time Differential Factor

Please pick an option number [2]: 2
```

   The system displays information like the following:

```
    SYSTEM TIME DIFFERENTIAL FACTOR = -4:00  (-14400 seconds).
    LOCAL SYSTEM TIME               = 22-SEP-1995 10:49:45.20.
```

2. Select option 1 to verify the TDF displayed or to enter a new one:

```
    Configuring the Time Differential Factor (TDF)

    Enter ? anytime for help

    [0]     Exit
    [1]     Set the Time Differential Factor
    [2]     Display the Time Differential Factor

Please pick an option number [2]: 1
```

   The system then displays the following information:

```
    The Time Differential Factor (TDF) is the difference between your
    system time and Coordinated Universal Time (UTC).  UTC is similar
    in most repects to Greenwich Mean Time (GMT).

    The TDF is expressed as hours and minutes, and should be entered
    in the hh:mm format.  TDFs for the Americas will be negative
    (-3:00, -4:00, etc.); TDFs for Europe, Africa, Asia and Australia
    will be positive (1:00, 2:00, etc.).
```

   The system displays the following question only if you set the time zone as well. However, some time zones do not have daylight saving time; if they do not, the system also does not display this question.

3. Answer Yes or No to the following question:

```
Is Daylight Savings time in effect? (Yes/No):
```

4. After the following prompt, either press Return to accept the displayed default or enter the correct TDF. (If you have not set the time zone, the system does not display a default TDF value.)

```
Enter the Time Differential Factor [-4:00]:
```

   The system then explains the need to modify the system time as well for season time changes:

```
    If this is a seasonal time change, it may also be necessary to
    modify the system time.  Generally, seasonal time changes result
    in adding 1:00 hour, or adding -1:00 hour to the system time.
```

5. To the following question, answer Yes if you need to modify the local system time or No if you do not:

   ```
   Do you wish to modify the local system time [N]:
   ```

   Answer No if you have already adjusted the system time using the SET TIME command.

   If you answer Yes, the system will lead you through a dialogue similar to the one in Section 3.6.4.

   If you answer No, the system next displays the new TDF:

   ```
   NEW SYSTEM TIME DIFFERENTIAL FACTOR = -4:00.
   ```

6. Answer Yes to confirm the displayed TDF or No if it is incorrect:

   ```
   Is this correct? [Y]:
   ```

   If you answer No, the system returns to step 1 in this section.

   If you answer Yes, the system displays both the TDF and the local system time.

   ```
   SYSTEM TIME DIFFERENTIAL FACTOR = -4:00  (-14400 seconds).
    LOCAL SYSTEM TIME               = 22-SEP-1995 10:52:37.36.
   ```

3.6.4 Adjusting for Daylight Saving Time and Standard Time

To adjust the local time to daylight saving or standard time, you can invoke the command procedure SYS$EXAMPLES:DAYLIGHT_SAVINGS.COM to do both of the following:

- Adjust the TDF
- Modify the local time

DAYLIGHT_SAVINGS.COM allows you to do either one of the following:

- Make the changes immediately. (Usually, however, you would use UTC$TIME_SETUP.COM to make changes immediately by answering Yes to Question 5 in Section 3.6.3.2.)
- Queue a batch job to make the changes at a future time. (This is the most common use of this command procedure.)

The following example of DAYLIGHT_SAVINGS.COM shows answers that cause the procedure to queue a batch job, DST_CHANGE, which will execute when the time changes from standard time to daylight saving time. Many of the questions are similar to those explained in Section 3.6.3.2.

In the example, the initial TDF value is -5:00. The local date and time are any time from the date in 1993 when the change to standard time was made, until 23-APR-1995:02:00, when the change to daylight saving time will be made.

```
$ SYS$EXAMPLES:DAYLIGHT_SAVINGS

        This procedure queues a batch job that changes the system time
        and system time differential around a daylight saving time
        change.  Press the question mark (?) key at any time for help;
        hit Control-C to exit.

        The Time Differential Factor (TDF) is the difference
        between your system time and Coordinated Universal Time (UTC).
        The difference is expressed in hh:mm format.  The Americas
        have negative offsets from UTC, while Europe, Africa, Asia
        and Australia have positive offsets from UTC.
```

```
       * Enter the Time Differential Factor: -4:00 Return

        If this is a seasonal time change, it may also be
        necessary to modify the system time.  Generally,
        seasonal time changes result in adding 1:00 hour,
        or adding -1:00 hour to the local time.

        * Do you wish to modify the local system time [N]: Y Return

        Enter the time value you would like  to add to
        the local time.  The value can be a positive or
        a negative (-hh:mm) value.

        * Enter the time value: +1:00 Return

        The process to modify your time zone offset and local
        time (if supplied) can occur now or in the future.
        Press Return to run the job now.

        * Enter the run time in the DD-MMM-YYYY:HH:MM:SS format: 23-apr-1995:02:00 Return
NEW SYSTEM TIME DIFFERENTIAL FACTOR = -4:00.
ADDING 1:00 TO THE LOCAL TIME.
JOB RUN TIME : 23-APR-1995:02:00

        * Continue? [Y]: Y Return
Job DST_CHANGE (queue SYS$BATCH, entry 2) holding until 23-APR-1995 02:00
Batch Job DST_CHANGE scheduled to run at 23-APR-1995:02:00
$
$!!The batch job DST_CHANGE will run on 23-Apr-1995 at 02:00
```

3.6.5 Setting Time in a VMScluster Environment

The TDF and the local time must be the same on all nodes in a VMScluster environment. You can use the System Management utility (SYSMAN) DO command to invoke the command procedure UTC$CONFIGURE_TDF.COM on one node in a cluster to do the following for one or more nodes in the cluster:

- Display the TDF
- Set or change the TDF
- Modify the local time

Note that UTC$CONFIGURE_TDF.COM is normally run by UTC$TIME_SETUP.COM when you select the TDF or BOTH option.

You specify the function SET or SHOW, the TDF value, and the local time modification as command procedure parameters. Note that you must express the TDF and the change to the local time in minutes format, not in hours and minutes *hh:mm* format, as you do if you use the command procedure interactively. For example, for +2:00, you would enter +120 or 120.

Examples

1. The following example changes the TDF clusterwide by +1:00 to 11:00 and also moves the local time ahead by 1 hour, as you would do if you were changing from standard time to daylight saving time. The initial TDF value is +10:00, and the local date and time are 12-MAR-1995:15:20.

```
$  RUN SYS$SYSTEM:SYSMAN

            SYSMAN>  SET ENVIRONMENT/CLUSTER

%SYSMAN-I-ENV, Current command environment:
         Clusterwide on local cluster
         Username SMITH will be used on nonlocal nodes

SYSMAN>  DO @SYS$MANAGER:UTC$CONFIGURE_TDF SET +660 +60
```

2. The following example displays the TDF for each node in the cluster:

```
$  RUN SYS$SYSTEM:SYSMAN
SYSMAN>  SET ENVIRONMENT/CLUSTER

%SYSMAN-I-ENV, Current command environment:
         Clusterwide on local cluster
         Username SMITH will be used on nonlocal nodes

SYSMAN> DO @SYS$MANAGER:UTC$CONFIGURE_TDF SHOW
```

   The system will display the TDF and the local time on each node in the cluster.

3. The following command procedure can be submitted as a batch job to run at a future time to do the following:

   - Change the TDF clusterwide by –1:00, to –4:30. (The initial TDF value is –3:30.)
   - Move the local time back by 1 hour 23-JAN1995.

```
$! TO_STANDARD_TIME.COM
$! Command procedure to change the TDF by -1:00 and also modify local time
$ RUN SYS$SYSTEM:SYSMAN
        SET ENVIRONMENT/CLUSTER
        DO @SYS$MANAGER:UTC$CONFIGURE_TDF SET -270 -60
        EXIT
```

   Note: Because UTC$CONFIGURE_TDF.COM accepts time in minutes, -4:30 is expressed as -((4 × 60) + 30) or -270; -1:00 is expressed as -60.

   The following command submits this command procedure as a batch job to run at 23-JAN1995:02:00:

```
$ SUBMIT/AFTER=23-JAN1995:02:00 TO_STANDARD_TIME.COM
```

3.7 System Dump Analyzer (SDA) Features

The following sections describe a new command, new features to existing commands, display changes, and other command changes for the System Dump Analyzer (SDA).

For more information, see the *OpenVMS Alpha Guide to Upgrading Privileged-Code Applications*, and the *OpenVMS Alpha System Dump Analyzer Utility Manual*.

3.7.1 New SET FETCH Command

The new command SET FETCH sets the default size of data manipulated by the EXAMINE and EVALUATE commands. It has no qualifiers.

3.7.2 Current Commands with New Features

The existing commands with new features are as follows:

3.7.2.1 SHOW CLUSTER

The command SHOW CLUSTER has the following new qualifier:

Qualifier	Meaning
/NODE=name	Displays cluster information on a particular VMScluster member node specified by its SCS node name.

3.7.2.2 SHOW CONNECTIONS

The command SHOW CONNECTIONS has the following two new qualifiers:

Qualifier	Meaning
/SYSAP=name	Displays all connection descriptor tables (CDTs) associated with the specified local SYSAP.
/NODE=name	Displays all connection descriptor tables (CDTs) associated with the specified remote SCS node name.

3.7.2.3 SHOW LAN

The command SHOW LAN has modified descriptions for the following:

Qualifier	Meaning
/CLIENT=name	Specifies that information be displayed for the specified client. The /CLIENT, /DEVICE, and /UNIT qualifiers are synonymous and mutually exclusive.
/DEVICE=name	Specifies that information be displayed for the specified device, unit, or client. The /CLIENT, /DEVICE, and /UNIT qualifiers are synonymous and mutually exclusive.
/UNIT=name	Specifies that information be displayed for the specified unit. The /CLIENT, /DEVICE, and /UNIT qualifiers are synonymous and mutually exclusive.

3.7.2.4 SHOW LOCK

The command SHOW LOCK has the following new qualifier:

Qualifier	Meaning
/CACHED	Displays locks that are no longer valid. Cached locks are not displayed in the other SHOW LOCK commands.

3.7.2.5 SHOW PAGE_TABLE

The command SHOW PAGE_TABLE has the following new qualifiers:

Qualifier	Meaning
/GPT	Specifies the portion of Page Table Space that maps the Global Page Table as the address range.
/PT	Specifies Page Table Space as the address range, as viewed in the context of the current process, or as viewed from system context if there is no current process.
/S0S1	Specifies S0 and S1 Space as the address range.
/S2	Specifies S2 Space as the address range.
/SPTW	Displays the contents of the System Page Table window. Level qualifiers are ignored.

3.7.2.6 SHOW PFN_DATA

The command SHOW PFN_DATA has the modified qualifier as follows:

Qualifier	Meaning
/ADDRESS=<PFN-entry-address>	Displays the PFN database entry at the address specified. The address specified is rounded to the nearest entry address so if you have an address that points to one of the fields of the entry, the correct database entry will still be found.

This command has a changed description and page frame number information as follows:

For each page frame number it displays, the SHOW PFN_DATA command lists information used in translating physical page addresses to virtual page addresses. The display has two lines of information. Table 3–3 shows the first line's fields; Table 3–4 shows the second line's fields.

Table 3–3 Page Frame Number Information—Line One Fields

Item	Contents
PFN	Page frame number.
DB ADDRESS	Address of PFN database entry for this page.
PT PFN	PFN of the page table page that maps this page.
BAK	Place to find information on this page when all links to this PTE are broken: typically, either an index into a process section table or the number of a virtual block in the paging file.
FLINK	Forward link within PFN database that points to the next physical page; this longword also acts as the count of the number of processes that are sharing this global section.
BLINK	Backward link within PFN database; also acts as an index into the working set list.
SWP/BO	Either a swap file page number or a buffer object reference count, depending on a flag set in the page state field.

(continued on next page)

Table 3–3 (Cont.) Page Frame Number Information—Line One Fields

Item	Contents
LOC	Location of the page within the system. It is one of the following eight types:

Location	Meaning
ACTIVE	Page is in a working set.
MDFYLST	Page is in the modified-page list.
FREELST	Page is in the free-page list.
BADLST	Page is in the bad-page list.
RELPEND	Release of the page is pending.
RDERROR	Page has had an error during an attempted read operation.
PAGEOUT	Page is being written into a paging file.
PAGEIN	Page is being brought into memory from a paging file.

FLAGS — Displays in text form the flags that are set in page state. Possible flags are:

Flag	Meaning
BUFOBJ	Set if any buffer objects reference this page.
COLLISION	Empty collision queue when page read is complete.
BADPAG	Bad page.
RPTEVT	Report event on I/O completion.
DELCON	Delete PFN when REFCNT=0.
MODIFY	Dirty page (modified).
UNAVAILABLE	PFN is unavailable. Most likely a console page.

Table 3–4 Page Frame Number Information—Line Two Fields

Item	Contents
Blank	
PTE ADDRESS	Virtual address of the page table entry that describes the virtual page mapped into this physical page.
Blank	
Blank	
Blank	
Blank	
REFCNT	Number of references being made to this page.

(continued on next page)

Table 3–4 (Cont.) Page Frame Number Information—Line Two Fields

Item	Contents
PAGETYP	Type of physical page. It is one of the following:

Page Type	Meaning
PROCESS	Page is part of process space.
SYSTEM	Page is part of system space.
GLOBAL	Page is part of a global section.
PPT(Ln)	Page is part of a process page table, where n is the page table level number.
GPGTBL	Page is part of a global page table.
GBLWRT	Page is part of a global, writable section.
UNKNOWN	Unknown.

Blank	

3.7.2.7 SHOW PROCESS

The command SHOW PROCESS has the following new qualifiers:

Qualifier	Meaning
/ADDRESS=pcb-address	Specifies the process control block (PCB) address of a process in order to display information about the process.
/INDEX=n	Displays the software and hardware context of the thread that is specified by the index of the software PCB into the system's PCB vector.
/RDE=id	Lists the information contained in the process region table for the specified region.
/SEMAPHORE	Displays the Inner Mode Semaphore for a multithreaded process.
/THREADS	Displays the software and hardware context of all the threads associated with the current process.
/WORKING_SET_LIST	Displays the working set list for the process.

3.7.2.8 SHOW RESOURCE

The command SHOW RESOURCE has the following new qualifier:

Qualifier	Meaning
/CACHED	Displays resource blocks that are no longer valid.

3.7.2.9 SHOW SUMMARY

The command SHOW SUMMARY has the following new qualifier:

Qualifier	Meaning
/THREAD	Displays information on all the current threads associated with the current process.

For more detailed information, see the *OpenVMS Alpha Guide to Upgrading Privileged-Code Applications* and the Bookreader version of the *OpenVMS Alpha System Dump Analyzer Utility Manual*.

3.7.3 Display Changes

The following shows display changes made to the SHOW CRASH and SHOW PFN_DATA commands.

SHOW CRASH

The SHOW CRASH command's exception display is changed to use the new 64-bit signal array. This change was made because the 32-bit signal array is often misleading when dealing with addresses outside the sign-extended 32-bit address spaces.

Fields Changed or Eliminated

Table 3–5 lists the displays that are changed to allow the displaying of 64-bit quantities. The fields for the following displays may have changed names or been eliminated.

Table 3–5 Display Changes

Display	Commands
Page table	SHOW PAGE_TABLE and SHOW PROCESS/PAGE_TABLES
Working set	SHOW PROCESS/WORKING_SET
Process header	SHOW PROCESS/PHD
Section table	SHOW PROCESS/PROCESS_SECTION_TABLE

SHOW PFN_DATA

The output of the SHOW PFN_DATA command, the PFN database display, has been converted to a two line format. This was necessary because of the expansion of virtual addresses from 32 bits to 64 bits, along with the need to display additional information.

Some of the fields of the SHOW PFN_DATA command have changed position, and new pieces of information appear for certain page types. Of particular attention are pages that formerly displayed PPGTBL in the PAGETYP column of the display. These pages are now shown as one of the following:

PHD
PPT(L1)
PPT(L2)
PPT(L3)

These correspond to the second column of status fields in the working set list display produced by the SHOW PROCESS/WORKING_SET_LIST command.

For more detailed information, see the *OpenVMS Alpha Guide to Upgrading Privileged-Code Applications*.

3.7.4 Other Command Changes

Page Table Requests

Certain page table display requests require SDA to traverse very large virtual address spaces in very small increments looking for valid page table entries. Particular examples are the SHOW PROCESS/PAGE_TABLES/P2, SHOW PAGE_TABLES/PT, and SHOW PAGE_TABLES/S2 commands, when used without parameters.

You may reduce or eliminate the pause by more narrowly specifying the address ranges of interest. Note also that this condition is more pronounced when analyzing running systems.

BUGCHECK Usability Changes

Changes have been made to BUGCHECK to attempt to increase the usability of dumps that were partially written when the system suffered some sort of catastrophic failure, reducing the number of instances where SDA signals an incomplete dump file.

This means, however, that dump files may not contain sections that the crash-time context might suggest they would. For instance, there might be no processes in the dump even though there were processes running at the time and there was room for them in the dump file.

BUGCHECK Transition Pages

Formerly, BUGCHECK forced pages of virtual memory that were in transition but not being read in from disk into validity at the time of a crash dump so that the pages could be written into the dump. To do this, it set the valid bit in the page table entries. This alteration was not previously visible. With the new address space architecture and the necessities it imposes on selective crash dumps, they are now visible.

To avoid the confusion inherent in having SDA display information that was not truly representative of the system state at the time of the crash, it was decided to change BUGCHECK to not alter the page table entries. All transition pages are treated as holes in address space, inaccessible by SDA.

For more detailed information, see the *OpenVMS Alpha Guide to Upgrading Privileged-Code Applications*.

3.7.5 SDA VAX Dump File Process

VAX

OpenVMS VAX SDA now uses RMS file access to process the dump file, instead of mapping the dump file into the working set. Some SDA commands, such as the SEARCH command, may be visibly slower because of this change.

Another affect of this change is that a value of 16,000 for the system parameter virtual page count VIRTUALPAGECNT should be sufficient to analyze any dump, even if a large number of symbols is read in.

For more detailed information, see the *OpenVMS Alpha System Dump Analyzer Utility Manual* on Bookreader. ♦

3.8 Warranted and Migration Support for VMScluster System Configurations

OpenVMS Alpha Version 7.0 and OpenVMS VAX Version 7.0 provide two levels of support for mixed-version and mixed-architecture VMScluster systems. These two support types are warranted and migration.

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

Migration support is a superset of the Rolling Upgrade support provided in earlier releases of OpenVMS and is available for mixes that are not warranted. Migration support means that Digital has qualified the versions for use together in configurations that are migrating in a staged fashion to a newer version of OpenVMS VAX or to OpenVMS Alpha. Problem reports submitted against these configurations will be answered by Digital. However, in exceptional cases Digital may request that you move to a warranted configuration as part of answering the problem.

Migration support will help customers move to warranted VMScluster version mixes with minimal impact on their cluster environments. Figure 3–7 shows the level of support provided for all possible version pairings.

Figure 3–7 VMScluster Version Pairings

	Alpha Version 6.2	*VAX Version 6.2*	*VAX Version 7.0*	*Alpha Version 7.0*
VAX Version 7.0	Migration	Migration	———	Warranted
Alpha Version 7.0	Migration	Migration	Warranted	———

ZK–7496A–GE

Note that Digital does not support the use of Version 7.0 with Version 6.1 (or earlier versions) in a VMScluster at a time. In many cases, mixing Version 7.0 with versions prior to Version 6.2 will successfully operate, but Digital cannot commit to resolving problems experienced with such configurations.

3.9 Printing Features

3.9.1 1024 Process Identifiers in Print Queuing Requests

Beginning with OpenVMS Version 6.2, users were allowed to have a maximum of 1024 identifiers associated with a process. However, users with more than 512 process identifiers could not use the queue system to print a file directly. In OpenVMS Version 7.0, the 512 identifier limit is increased to 1024 for the Version 7.0 queue system.

3.9.2 New Qualifier for Print Queues

By default, the print queues give an initial form feed to insure the paper is at the top of the page before printing begins.

In this release of OpenVMS, a new qualifier, /NO_INITIAL_FF, was added to the DCL commands INITIALIZE/QUEUE, SET QUEUE, and START/QUEUE to control the initial form feed that is sent to the print queue. Using this qualifier suppresses the form feed.

For more information, see *OpenVMS DCL Dictionary: N–Z.*

4
Programming Features

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

4.1 OpenVMS Alpha 64-Bit Addressing Support (Alpha Only)

Alpha

OpenVMS Alpha Version 7.0 provides support for 64-bit virtual addressing, which makes the 64-bit virtual address space defined by the Alpha architecture available to the OpenVMS Alpha operating system and to application programs. The 64-bit addressing features allow processes to map and access data beyond the limits of 32-bit virtual addresses. Both process-private and system virtual address space now extend beyond 2 GB.

In addition to the dramatic increase in virtual address space, OpenVMS Alpha 7.0 significantly increases the amount of physical memory that can be used by individual processes.

Many tools and languages supported by OpenVMS Alpha (including the Debugger, run-time library routines, and DEC C) are enhanced to support 64-bit virtual addressing. Input and output operations are performed directly to and from the 64-bit addressable space by means of RMS services, the $QIO system service, and most of the device drivers supplied with OpenVMS Alpha systems.

Underlying this are new system services, which allow an application to allocate and manage the 64-bit virtual address space that is available for process private use.

Nonprivileged programs may optionally be modified to exploit 64-bit addressing support. OpenVMS Alpha 64-bit virtual addressing does not affect nonprivileged programs that are not explicitly modified to exploit 64-bit support. Binary and source compatibility of existing nonprivileged programs is guaranteed.

For more information about OpenVMS Alpha 64-bit virtual addressing features, see the *OpenVMS Alpha Guide to 64-Bit Addressing*. ♦

4.2 OpenVMS Debugger

The following sections describe new features for the OpenVMS Debugger.

4.2.1 Debugging Optimized Code (Alpha Only)

Alpha

The ability to debug code compiled with full optimization has been significantly improved on OpenVMS Alpha systems. You can compile your program with the /DEBUG and /OPTIMIZE qualifiers and debug the resulting executable file.

In order to take advantage of the new features that improve the ability to debug optimized code, you will need a new version of your language compiler. For definitive information about the necessary version of your compiler, please see your compiler release notes or other compiler documentation.

Two areas of support for debugging optimized code have been implemented:

- Split-lifetime support
- A new semantic stepping mode

Note that about one-third more disk space is needed for debugging optimized code, to accommodate the increased image size.

Split-Lifetime Variables

In compiling with optimization, the compiler sometimes performs split-lifetime analysis on a variable, "splitting" it into several independent subvariables that can be independently allocated. The effect is that the original variable can be thought to reside in different locations at different points in time — sometimes in a register, sometimes in memory, and sometimes nowhere. It is even possible for the different subvariables to be simultaneously active.

Previously on Alpha, the debugger only reported the first subvariable of a split-lifetime variable, and then only when that instance was in scope. With this version, the debugger now can report all subvariables of a split-lifetime variable automatically by means of the EXAMINE command.

Split-lifetime analysis applies only to scalar variables and parameters. It does not apply to arrays, records, structures, or other aggregates.

EXAMINE/DEFINITIONS Command

For a split-lifetime variable, the EXAMINE command on Alpha systems not only displays the value of the active lifetime, it also displays the lifetime's definition points. The definition points are places where the lifetime could have received an initial value (if there is only one definition point, then that is the only place.) The /DEFINITIONS qualifier enables you to customize the number of definition points displayed. This is useful for determining where a variable obtained an inappropriate value.

Semantic Events

One of the problems of stepping through optimized code is that the apparent source program location "bounces" back and forth with the same line often appearing again and again. Indeed, sometimes the forward progress in STEP LINE mode averages barely more than one instruction per STEP command.

This problem is addressed through annotating instructions that are semantic events. Semantic events are important for two reasons:

- They represent the points in the program where the "effects" of the program actually happen.
- These effects tend to happen in an order that remains close to the source order of events in the program.

A semantic event is one of the following:

- Data event — An assignment to a user variable
- Control event — A control flow decision, with a conditional or unconditional transfer of control, other than a call
- Call event — A call (to a routine that is not stepped over) or a return from a call

It is important to understand that not every assignment, transfer of control, or call is necessarily a semantic event. The major exceptions are as follows:

- When two instructions are required to assign to a complex or X_floating value, only the first instruction is treated as an event point.
- When there are multiple branches that are part of a single higher-level construct, such as a decision tree of branches that implement a case or select construct, then only the first is treated as an event point.
- When a call is made to a routine that is a compiler-specific helper routine, such as a call to OTS$MOVE, which handles certain kinds of string or storage copy operations, the call is not considered an event point and control will not stop at the call.
- When there is more than one semantic event in a row with the same line number, then only the first is used.

SET STEP SEMANTIC_EVENT Command

The SET STEP SEMANTIC_EVENT command establishes the default stepping mode as semantic.

STEP/SEMANTIC_EVENT Command

STEP/SEMANTIC_EVENT, or simply STEP when semantic mode is in effect, causes a breakpoint to be set at the next semantic event. Execution proceeds to that next event. Parts of any number of different lines/statements may be executed along the way, without interfering with progress. When the semantic event is reached (that is, when the instruction associated with that event is reached but not yet executed), execution is suspended (similarly to reaching the next line when STEP/LINE is used).♦

4.2.2 Internationalization Features

For Asian users, the DECwindows Motif interface to the debugger as well as the command line and screen mode interfaces to the debugger can be used with multibyte characters.

Logical Names for Internationalization

If you use the screen (character-cell) mode, you enable country-specific features by defining logical names, as follows:

- DBG$SMGSHR — For specifying the Screen Management (SMG) shareable image. The debugger uses the SMG shareable image in its implementation of screen mode. Asian variants of the SMG shareable image handle multibyte characters. Hence, if an Asian variant of SMG is used by the debugger, the screen mode interface to the debugger will be able to display and manipulate multibyte characters.

  Define the DBG$SMGSHR logical name as follows:

  ```
  $  DEFINE/JOB DBG$SMGSHR <name_of_Asian_SMG>
  ```

  <name_of_Asian_SMG> varies according to the variants of Asian OpenVMS. For example, the name of the Asian SMG in Japanese OpenVMS is JSY$SMGSHR.EXE.

- SMG$DEFAULT_CHARACTER_SET — For the Asian SMG and multibyte characters. This logical need only be defined if DBG$SMGSHR has been defined. Please refer to the User's Guide to Asian or Japanese Screen Management Routines for details on how to define this logical.

You can modify your resource file, VMSDEBUG.DAT, so that characters other than those in the ISO Latin-1 character set can be displayed in the DECwindows Motif interface. For information, see Section 4.2.5.

4.2.3 SHOW CALLS and SHOW STACK Commands and Null Frame Procedures (Alpha Only)

Alpha

The OpenVMS Debugger Version 7.0 provides an improved display in response to the debugger SHOW CALLS and SHOW STACK commands. The display now includes the frames of null frame procedures. (Null frame procedures do not establish their own context on the stack, but rather, use the context of the routine that called them.)

For more information on null frame procedures, see the *OpenVMS Calling Standard*.♦

4.2.4 CALL Command and Floating-Point Parameters

The CALL command now allows you to pass floating-point parameters by value. They are passed in F_floating format.

The CALL command assumes that the floating-point value being passed is in F_ floating format. Passing a floating-point value in a format other than F_floating is not supported, and has unpredictable consequences.

This feature, previously supported on OpenVMS VAX systems, is now also supported on OpenVMS Alpha systems.

4.2.5 Customization Features for the DECwindows Motif Interface

The OpenVMS Debugger Version 7.0 provides improved user customization features for the DECwindows Motif interface:

- New version of the resource file, VMSDEBUG.DAT
- Customization-related new items in the Source View menu bar Options pulldown menu
- Customizable font definitions in all but the Heap Analyzer windows

Resource File

The new version of the resource file, VMSDEBUG.DAT, includes statements for customizing the following interface features:

- Source, Control, and Instruction view geometry
- Editor and Source Browser window geometry
- Line Number and Address display in the Source and Instruction views
- Heights of all window panes in both the Source and Control views
- User-defined push button definitions and labels or pixmaps
- Fonts for most windows, with the option of specifying a DebugDefault.Font resource to use a single font for all views (when you comment out the view-specific font resources), or commenting out all font resource specifications to use the system default font
- Startup initial state (visible or iconified) for most windows
- Colors for elements of the Source and Instruction views and Editor
- User-defined keypad definitions

- Command echo
- Title bar label format

VMSDEBUG.DAT now contains extensive comments regarding each category of customization and a default style that illustrates most of the customizable features.

VMSDEBUG.DAT is installed in the DECW$SYSTEM_DEFAULTS directory, and, if you permit, in your own DECW$USER_DEFAULTS directory (typically SYS$LOGIN). When you invoke the Motif interface, the debugger identifies any old resource file formats, and requests that you upgrade to the new format by executing a Save Options or Restore Default Options command from the Options menu.

The debugger does not force you to upgrade because some users have previously customized their resource files and may not wish to supersede them. However, it is recommended that the features in the new version be tried. The debugger will not purge older resource file versions, and you can merge previous customizations into the new resource file or fall back to your previous resource file version by simply deleting the newer file.

System managers can modify the system default resource file (DECW$SYSTEM_DEFAULTS:VMSDEBUG.DAT), which can propagate common features or styles to all end users.

Note

Do not alter the "DebugVersion" statement, as the interface has hard-coded version expectations.

Menu Items

The new Source View Options menu items are as follows:

- Save Options—Stores the current geometry of visible windows and various interface states in the user's resource file (DECW$USER_DEFAULTS:VMSDEBUG.DAT). It does not purge older resource file versions. You can easily fall back to a previous version by simply deleting the newer file. The format of the saved resource file is a terse collection of statements with no comments. If you want to retain comments or customizations in your previous resource file version, you should review and merge it with the newly saved VMSDEBUG.DAT.
- Restore Default Options—Supersedes the user-specific resource file with the system default resource file, copied from DECW$SYSTEM_DEFAULTS:VMSDEBUG.DAT to DECW$USER_DEFAULTS:VMSDEBUG.DAT. The new version of the file becomes active the next time you invoke the debugger's DECwindows Motif interface. This is the simplest way to upgrade to the newer resource file. Previous resource file versions are not purged, and you can review and merge features of the new and previous resource files, or return to your previous style by simply deleting the newer file.
- Edit Options File—Loads and displays your resource file (DECW$USER_DEFAULTS:VMSDEBUG.DAT) in the Debug Editor. Then you can easily review and customize the interface style. If you do not have a DECW$USER_DEFAULTS:VMSDEBUG.DAT file, and you select EDIT OPTIONS FILE, you will get an empty editor with a message saying the file was not found.

Customizing Fonts

You can modify your resource file, DECW$USER_DEFAULTS:VMSDEBUG.DAT, to use the system's default fonts by commenting out all font resource lines in the file. You can edit the resource file from within the debugger using the new Edit Options File command from the Options menu, or you can edit the file using a separate editor. The system default font will be used the next time you start the debugger.

4.2.6 Editor File Menu Items in the DECwindows Motif Interface

The new Editor File menu items are as follows:

- Refresh File—Loads and displays the latest version of the file that is currently in the selected editor buffer. This feature is convenient for reviewing results files during debugging.
- Close File—Removes the currently selected edit buffer from the display and menu, closes the file, and frees the associated memory resources. If the edit buffer has been modified, the user is prompted to save or ignore changes before closing the file.

4.2.7 Command/Message View Popup Menu Items in the DECwindows Motif Interface

The new Command/Message View Popup menu items are as follows:

- Clear Command Window—Clears the entire Command/Message View, leaving only the current command line prompt.
- Clear Command Line—Clears the current command line in the Command /Message View.

4.2.8 Documentation Changes

The *OpenVMS Debugger Manual* has been revised for Version 7.0. Complete reference information on debugger commands, formerly available only in online help, has been added to the manual as a command dictionary. In addition, information about the new debugger features has been added.

The command dictionary is also available in printable form in the following file:

SYS$HELP:DBG$HELP.PS

The .TXT format of this file is no longer provided.

4.3 Heap Analyzer Support (Alpha Only)

Alpha

The Heap Analyzer provides a graphical representation of memory use in real time. By studying this representation, you can quickly identify areas in your application where memory usage and performance can be improved. For example, you might notice allocations that are made too often, memory blocks that are too large, evidence of fragmentation, or memory leaks.

Once you have located an area of interest, you can request an enlarged, more detailed, or altered view. If you choose, you can see traceback or statistical information associated with an allocation. You can then display the source code associated with a particular allocation, and correlate memory events with application code.

You can invoke the Heap Analyzer in one of the following ways:

- From the DECwindows Motif Kept Debugger (GUI):

  Choose the Run Image or Rerun Same items from the File menu on the Command/Message View. When a dialog box appears, indicate the program you wish to execute and click the Heap Analyzer toggle button.

- From the DECwindows Motif Kept Debugger (Command Entry) or a DECterm emulating a terminal screen:

  At the DBG> prompt within the DECwindows Motif Kept Debugger, issue the RUN or RERUN command with the /HEAP_ANALYZER qualifier.

- From the DCL command line in a DECterm window:

  At the DCL prompt ($), issue the following command, and then execute your program with the RUN/NODEBUG command:

```
$ DEFINE/USER LIBRTL SYS$LIBRARY:LIBRTL_INSTRUMENTED
```

Note that the Heap Analyzer does not work on programs linked with the /NODEBUG qualifier on OpenVMS Alpha systems. (On OpenVMS VAX systems, the Heap Analyzer does work on programs linked with the /NODEBUG qualifier, although the traceback information displayed will be minimal.)

The Heap Analyzer, previously documented for OpenVMS VAX systems, is now also supported on OpenVMS Alpha Version 7.0. For a complete description of the feature, see the *OpenVMS Debugger Manual*. ♦

4.4 DECthreads Features

The following sections describe new DECthreads features.

4.4.1 DECthreads Implements Final POSIX 1003.1c Standard Style Interface

As of this release, the DECthreads library (PTHREAD$RTL.EXE) provides an implementation of the POSIX 1003.1c standard as approved by the IEEE standards board in June 1995 (IEEE Std 1003.1c-1995, POSIX System Application Program Interface). The new POSIX (pthread) style interface supported with DECthreads is the most portable, efficient, and powerful programming interface for a multithreaded environment. These interfaces are defined by <pthread.h>. For more detailed information, see the *Guide to DECthreads*.

4.4.2 Thread Independent Services (TIS) Interface

This release introduces the Thread Independent Services (TIS) application programming interface (CMA$TIS_SHR.EXE). TIS provides services that assist with the development of thread-safe libraries.

Thread synchronization may involve significant run-time cost, which is undesirable in the absence of threads. TIS enables thread-safe libraries to be built that are both efficient in the nonthreaded environment, yet provide the necessary synchronization in the threaded environment.

When DECthreads is not active within the process, TIS executes only the minimum steps necessary: code running in a nonthreaded environment is not burdened by the run-time synchronization that is necessary when the same code is run in a threaded environment. When DECthreads is active, the TIS functions provide the necessary thread-safe synchronization. For more detailed information, see the *Guide to DECthreads*.

4.5 DELTA/XDELTA Support for Debugging Multithreaded Applications (Alpha Only)

Alpha

To support the debugging of multithreaded applications, the capability of displaying a thread ID at a breakpoint has been added to DELTA. When you reach a breakpoint in a multithreaded application, DELTA displays the thread ID and stops the execution of all other threads. When you reach a breakpoint in a single-threaded application, the display and behavior is the same as in the past; DELTA displays the address and stops program execution.

In the following example, a breakpoint is set with `30000;B` and is followed by the `;P` (Proceed from Breakpoint) command. The breakpoint is taken. Because it is a multithreaded application, the thread ID is included in the display.

```
30000;B ;P
Brk 1 at 30000 on Thread 12
00030000!  LDA  SP,#XFF80(SP)♦
```

4.6 Global Section Limit Increased on OpenVMS Alpha Version 7.0

Alpha

In OpenVMS Alpha Version 7.0, the global section limit has been increased from 3,276 to 65,535. ♦

VAX

On OpenVMS VAX Version 7.0, the global section limit remains 4,095. ♦

4.7 High-Performance Sort/Merge Utility—SOR$ Routines (Alpha Only)

Alpha

An optional high-performance Sort/Merge utility takes advantage of the Alpha architecture to provide better performance for most sort and merge operations. Currently, the high-performance Sort/Merge utility supports a subset of the SORT/MERGE SOR$ routines.

This section describes the callable interface to the high-performance Sort/Merge utility by way of the SOR$ routines. See the Bookreader version of the *OpenVMS Utility Routines Manual* for information about the SOR$ routines. Refer to Section 2.2 for information about using the high-performance Sort/Merge utility from the command line.

See the *OpenVMS Version 7.0 Release Notes* for information related to problems and restrictions associated with this release of the high-performance Sort/Merge utility.

Selecting High-Performance Sort

Use the SORTSHR logical to select the high-performance Sort/Merge utility. Define SORTSHR to point to the high-performance sort executable image in SYS$LIBRARY, as follows:

```
$ define sortshr sys$library:hypersort.exe
```

To return to the Sort/Merge utility, deassign SORTSHR. The Sort/Merge utility is the default if SORTSHR is not defined.

4.7 High-Performance Sort/Merge Utility—SOR$ Routines (Alpha Only)

SOR$ Routine Behavior

The behavior of the SOR$ routines for the high-performance Sort/Merge utility is the same as for SORT/MERGE except as shown in Table 4–1.

Table 4–1 High-Performance Sort/Merge: Differences in SOR$ Routine Behavior

Feature	High-Performance Sort/Merge Behavior
Output file organization	Indexed sequential output file organization is not supported.[1] Do not specify the SOR$PASS_FILES routine **org** argument as FAB$C_IDX or the **rfm** argument as FAB$C_VFC.
Work files	Permissible values of the SOR$BEGIN_SORT **work_files** argument range from 1 through 255. By default, the high-performance Sort/Merge utility creates one temporary work file.
Input file size	If you do not specify an input file size in the SOR$BEGIN_SORT **file_alloc** argument, the high-performance Sort/Merge utility determines a default based on the size of the input file, or if input is not from files, on available memory.
Specification files	The SOR$SPEC_FILE routine is not supported.[1]
Key data types	DSCK_DTYPE_O, DSCK_DTYPE_OU, DSC$K_DTYPE_H, and DSC$K_DTYPE_NZ are not valid key data types in the SOR$BEGIN_MERGE or SOR$BEGIN_SORT **key_buffer** argument.[1]
Key data types not normally supported by SORT/MERGE	The SOR$DTYPE routine is not supported.[1] Data types that would otherwise be specified using SOR$DTYPE include extended data types and the National Character Set (NCS) collating sequences.
Internal sorting processes	Only the record sort process is supported. You can specify the SOR$BEGIN_SORT routine **sort_process** argument as SOR$GK_RECORD or omit the argument. The SORGK_TAG, SORGK_ADDRESS, and SOR$GK_INDEX values are not supported for the **sort_process** argument.[1]
Statistical summary information	The SOR$STAT routine is not supported.[1]
User-supplied action routines	The following user-supplied action routines are not supported for either SOR$BEGIN_MERGE or SOR$BEGIN_SORT.[1] You must provide a placeholder comma (,) in the argument list if other arguments follow the customary position of the **user_compare** or **user_equal** argument. **user_compare** Compares records to determine their sort or merge order. **user_equal** Resolves the sort or merge order when records have duplicate keys.

[1]Implementation of this feature is deferred to a future OpenVMS Alpha release.

If you attempt to use an unsupported capability, the high-performance Sort/Merge utility generates an error. High-performance Sort/Merge adds the following condition value to those listed for SORT/MERGE:

SOR$_NYI	Attempt to use a feature that is not yet implemented.♦

4.8 Kernel Threads (Alpha Only)

Alpha

With the implementation of kernel threads, OpenVMS Alpha V7.0 provides new advantages and some changed features to the operating system.

The following sections present this information.

4.8.1 Kernel Threads Advantages

Kernel threads on OpenVMS Alpha enables multiple execution contexts within a process, allowing more than one application thread to be executing at the same time. These execution contexts are called kernel threads. Kernel threads allows a multithreaded application to have a thread executing on every CPU in a multiprocessor system. Kernel threads also allows a threaded application to take advantage of multiple CPUs in a symmetric multiple processing (SMP) system.

By using kernel threads as a programming model, the application programmer can gain the following advantages in a program:

- More modular code
- Simpler application design and maintenance
- Independent flows of execution in parallel on multiple CPUs
- Better use of available CPU resources through parallel execution

4.8.2 New Kernel Threads Features

The OpenVMS Alpha operating system implements two new outstanding features:

- Multiple execution contexts within a process
- Efficient use of the OpenVMS and DECthreads schedulers

4.8.2.1 Multiple Execution Contexts Within a Process

Before the implementation of kernel threads, the scheduling model for OpenVMS was per process. The only scheduling context was the process itself; that is, only one execution context per process. Since a threaded application could create thousands of threads, many of these threads could potentially be executing at the same time. But because OpenVMS processes had only a single execution context, in effect only one of those application threads was running at any one time. If this multithreaded application was running on a multiprocessor system, the application could not make use of more than a single CPU.

After the implementation of kernel threads, the scheduling model allows for multiple execution contexts within a process; that is, more than one application thread can be executing concurrently. These execution contexts are called kernel threads. Kernel threads allows a multithreaded application to have a thread executing on every CPU in a multiprocessor system. Kernel threads also allows a threaded application to take advantage of multiple CPUs in a symmetric multiple processing (SMP) system.

4.8.2.2 Efficient Use of the OpenVMS and DECthreads Schedulers

It is the function of the user mode thread manager to schedule individual user mode application threads. On OpenVMS, DECthreads is the user mode threading package of choice. Before the implementation of kernel threads, DECthreads multiplexed user mode threads on the single OpenVMS execution context—the process. DECthreads implemented parts of its scheduling by using a periodic timer. If the AST executed and the thread manager gained control, the thread manager could then select a new application thread for execution. But because the thread manager could not detect that a thread had entered an OpenVMS wait state, the entire application blocked until that periodic AST was delivered. That resulted in a delay until the thread manager regained control and could schedule another thread. Once the thread manager gained control, it could schedule a previously pre-empted thread unaware that the thread was in a wait state. The lack of integration between the OpenVMS and DECthreads schedulers could result in wasted CPU resources.

After the implementation of kernel threads, the scheduling model provides for scheduler callbacks. A scheduler callback is an upcall from the OpenVMS scheduler to the thread manager whenever a thread changes state. This upcall allows the OpenVMS scheduler to inform the thread manager that the current thread is stalled and that another thread should be scheduled. Upcalls also inform the thread manager that an event a thread is waiting on has completed. With kernel threads, the two schedulers are better integrated, minimizing application thread scheduling delays.

For more detailed information about kernel threads, see the *OpenVMS Alpha Guide to Upgrading Privileged-Code Applications*. ♦

4.9 Linking for Different Architectures

It is possible to create OpenVMS Alpha images on an OpenVMS VAX system and to create OpenVMS VAX images on an OpenVMS Alpha system. To do this, you must mount a system disk of the target architecture and make it accessible on the system where the link is to occur. Also, you must assign several logical names to point to portions of the target architecture disk.

Table 4–2 lists the logical names and the conditions of their use.

Table 4–2 Logical Names for Cross-Architecture Linking

Logical Name	Description
ALPHA$LIBRARY	The linker uses this logical name when creating an OpenVMS Alpha image to locate the target system's shareable images and system libraries.
VAX$LIBRARY	The linker uses this logical name when creating an OpenVMS VAX image on an OpenVMS Alpha computer to locate the target system's shareable images and system libraries.
SYS$LIBRARY	The linker uses this logical name when creating an OpenVMS VAX image on an OpenVMS VAX computer to locate the target system's shareable images and system libraries.

(continued on next page)

Table 4–2 (Cont.) Logical Names for Cross-Architecture Linking

Logical Name	Description
ALPHA$LOADABLE_ IMAGES	The linker uses this logical when creating an OpenVMS Alpha image to locate the target system's base image SYS$BASE_ IMAGE.EXE when the /SYSEXE qualifier is in the link command line.

The /ALPHA and /VAX qualifiers control which architecture an image is built for:

- When you specify /ALPHA, the linker creates an OpenVMS Alpha image using the OpenVMS Alpha libraries and OpenVMS Alpha images from the target system disk that the logicals ALPHA$LIBRARY and ALPHA$LOADABLE_IMAGES point to. When you link on an OpenVMS Alpha system, these logical names initially point to the current system's libraries and images. The qualifier /ALPHA is the default on OpenVMS Alpha systems.
- When you specify /VAX on an OpenVMS Alpha system, the linker creates an OpenVMS VAX image using the OpenVMS VAX libraries and OpenVMS VAX images from the target system disk that the logical VAX$LIBRARY points to. On an OpenVMS VAX system, you create VAX images by using the OpenVMS VAX libraries and OpenVMS VAX images that the logical SYS$LIBRARY points to. The qualifier /VAX is the default on OpenVMS VAX systems.

The next two sections provide reference information about the LINK command qualifiers /ALPHA and /VAX.

4.9.1 /ALPHA Qualifier

The /ALPHA qualifier directs the linker to produce an OpenVMS Alpha image. The default action, when neither /ALPHA nor /VAX is specified, is to create an OpenVMS VAX image on an OpenVMS VAX system and to create an OpenVMS Alpha image on an OpenVMS Alpha system.

Format

```
/ALPHA
```

Qualifier Values

None.

Description

This qualifier is used to instruct the linker to accept OpenVMS Alpha object files and library files to produce an OpenVMS Alpha image.

You must inform the linker where OpenVMS Alpha system libraries and shareable images are located with the logical names ALPHA$LOADABLE_ IMAGES and ALPHA$LIBRARY. On an OpenVMS Alpha system, these logicals are already defined to point to the correct directories on the current system disk. On OpenVMS VAX, you must define these logical names so that they translate to the location of an OpenVMS Alpha system disk residing on the system where the Alpha linking is to occur.

Example

```
$ DEFINE ALPHA$LIBRARY DKB100:[VMS$COMMON.SYSLIB]
$ DEFINE ALPHA$LOADABLE_IMAGES DKB100:[VMS$COMMON.SYS$LDR]
$ LINK/ALPHA ALPHA.OBJ
```

This example, which is performed on an OpenVMS VAX system, shows the definition of logical names to point to the appropriate areas on an OpenVMS Alpha system disk mounted on device DKB100. The qualifier /ALPHA tells the linker to expect the object file, ALPHA.OBJ, to be an OpenVMS Alpha object file and to link it using the OpenVMS Alpha libraries and images on DKB100, if necessary.

4.9.2 /VAX Qualifier

The /VAX qualifier directs the linker to produce an OpenVMS VAX image. The default action, when neither /ALPHA nor /VAX is specified, is to create an OpenVMS VAX image on an OpenVMS VAX system and to create an OpenVMS Alpha image on an OpenVMS Alpha system.

Format

```
/VAX
```

Qualifier Values

None.

Description

This qualifier is used to instruct the linker to accept OpenVMS VAX object files and library files to produce an OpenVMS VAX image.

You must inform the linker where OpenVMS VAX system libraries and shareable images are located. On an OpenVMS VAX system, you use the logical name SYS$LIBRARY to do this. On an OpenVMS Alpha system, you use the logical name VAX$LIBRARY to do this. Therefore, if the link is to occur on an OpenVMS Alpha system, you must define the logical VAX$LIBRARY so that it translates to the location of an OpenVMS VAX system disk residing on the system where the VAX linking is to occur.

Example

```
$ DEFINE VAX$LIBRARY DKB200:[VMS$COMMON.SYSLIB]
$ LINK/VAX VAX.OBJ
```

This example, performed on an OpenVMS Alpha system, shows the definition of the logical name VAX$LIBRARY to point to an OpenVMS VAX system disk mounted on device DKB200 in the appropriate area. The qualifier tells the linker to expect the object file, VAX.OBJ, to be an OpenVMS VAX object file and to link it using the OpenVMS VAX libraries and images on DKB200, if necessary.

4.10 New /ALPHA Qualifier for Command Definition, Library, and Message Utilities

A new /ALPHA qualifier has been added to the Command Definition, Library, and Message utilities. In addition, the behavior of the /VAX qualifier has been modified in each of these utilities. For related information, see Section 4.9.

For detailed information about the new /ALPHA qualifier and changes to the /VAX qualifier, see the *OpenVMS Command Definition, Librarian, and Message Utilities Manual*.

4.11 New LAT Item Codes (Alpha Only)

Alpha

Table 4–3 shows the LAT node entity item codes.

Table 4–3 LAT Node Entity Item Codes

Item Code	Meaning
LAT$_ITM_LARGE_ BUFFERS	Boolean used to indicate whether or not the LAT software is using large packet support by default.
LAT$_ITM_ ANNOUNCEMENTS	Boolean used to indicate whether or not the LAT software is transmitting LAT service advertisement messages.

Table 4–4 shows the LAT port entity item code.

Table 4–4 LAT Port Entity Item Code

Item Code	Meaning	
LAT$_ITM_PORT_STATE	Current port state. Possible values are:	
	LAT$C_PT_STATE_ INACTIVE	Port is inactive.
	LAT$C_PT_STATE_ CONNECTING	Port connection in progress but not complete.
	LAT$C_PT_STATE_ACTIVE	Port has active LAT connection.
	LAT$C_PT_STATE_ DISCONNECTING	Port LAT connection in process of terminating.

For more detailed information, see the Bookreader version of the *OpenVMS I/O User's Reference Manual*. ♦

4.12 Mail Utility Features

This section describes the following new features for the Mail utility:

- The signature file user profile entry field
- Item codes to implement the signature file
- Mail routines that use these item codes

4.12.1 Signature File User Profile Entry Field

The Mail utility maintains an indexed data file VMSMAIL_PROFILE.DATA that serves as a systemwide database of **user profile** entries. A user profile entry is a record that contains data describing a Mail user's default processing characteristics and whose primary key is the user name. The following table shows information about the new user profile entry field:

Field	Function
Signature file	Text file that is automatically appended to the end of the body of a mail message

4.12.2 Input Item Codes for the Signature File in the Send Context

The new input item codes for the send context are as follows:

Item Code	Function
MAIL$_SEND_SIGFILE	Specifies a full OpenVMS file specification of the signature file to be used in the message.
MAIL$_SEND_NO_SIGFILE	Specifies that no signature file be used.

4.12.3 Input Item Codes for the Signature File in the User Context

The new input item codes for the user context are as follows:

Item Code	Function
MAIL$_USER_SET_SIGFILE	Specifies a signature file specification for the specified user.
MAIL$_USER_SET_NO_SIGFILE	Clears a signature file field for the specified user.

4.12.4 Output Item Code for the Signature File in the User Context

The new output item code for the user context is as follows:

Item Code	Function
MAIL$_USER_SIGFILE	Returns the default signature file specification.

4.12.5 MAIL$SEND_BEGIN Routine Input Item Codes

The following sections describe input item codes for the Mail routine MAIL$SEND_BEGIN.

4.12.5.1 MAIL$_SEND_SIGFILE and MAIL$_SEND_NO_SIGFILE

Note that you must specify only one of these item codes. An error is generated if you specify both item codes. MAIL$_SEND_SIGFILE specifies the full OpenVMS file specification of the signature file to be used in the message. The default file specification used for a signature file is the user mail directory specification and .SIG as the file type. The buffer address field of the item descriptor points to a buffer that contains a character string 0 to 255 characters long.

Specify a value from *0* to *255* in the **buffer length** field of the item descriptor.

The Boolean item code MAIL$_SEND_NO_SIGFILE specifies that no signature file be used during message construction.

4.12.6 MAIL$USER_BEGIN and MAIL$USER_GET_INFO Routines Output Item Code

The following is an output item code for the Mail routine MAIL$USER_BEGIN and MAIL$USER_GET_INFO.

4.12.6.1 MAIL$_USER_SIGFILE

When you specify MAIL$_USER_SIGFILE, MAIL$USER_BEGIN returns the default signature file specification. The **buffer address** field of the item descriptor points to a buffer that receives a character string 0 to 255 characters long.

Specify a value from *0* to *255* in the **buffer length** field of the item descriptor.

4.12.7 MAIL$USER_SET_INFO Routine Input Item Codes

The following are input item codes for the Mail routine MAIL$USER_SET_INFO.

4.12.7.1 MAIL$_USER_SET_SIGFILE and MAIL$_USER_SET_NO_SIGFILE

MAIL$_USER_SET_SIGFILE specifies a signature file specification for the specified user. The **buffer address** field of the item descriptor points to a buffer that contains a character string 0 to 255 characters long.

Specify a value from *0* to *255* in the **buffer length** field of the item descriptor.

The Boolean item code MAIL$_USER_SET_NO_SIGFILE clears the signature file field for the specified user.

Specify the value *0* in the **buffer length** and **buffer address** fields of the item descriptor.

4.13 New STARLET Definitions for C (Alpha Only)

Alpha

As of OpenVMS Alpha Version 7.0, SYS$LIBRARY:SYS$STARLET_C.TLB (or STARLET) provides C function prototypes for system services, as well as new and enhanced data structure definitions. The new definitions are more consistent with the OpenVMS C language coding conventions and definitions (typedefs) used in SYS$LIBRARY:SYS$LIB_C.TLB.

In order to maintain source compatibility for existing users of STARLET.H, the ''old style'' function declarations and definitions are still provided by default. To take advantage of the new system service function prototypes and type definitions, you must explicitly enable them.

You can define the _NEW_STARLET symbol with a DEC C command line qualifier or include the definition directly in your source program. For example:

- Define the _NEW_STARLET symbol with the DEC C command line qualifier as follows:

  ```
  /DEFINE=(__NEW_STARLET=1)
  ```

  or

- Define the _NEW_STARLET symbol in your C source program before including the SYS$STARLET_C.TLB header files:

  ```
  #define __NEW_STARLET 1

  #include  <starlet.h>
  #include  <vadef.h>
  ```

The system service function prototypes will be documented in a future release of OpenVMS Alpha. To see the currently available system service function prototypes in STARLET.H, you can use the Librarian utility as shown in the following example:

```
$ LIBRARY/OUTPUT=STARLET.H SYS$LIBRARY:SYS$STARLET_C.TLB/EXTRACT=STARLET
```

The following example shows a new system service function prototype as it is defined in STARLET.H:

```
#pragma __required_pointer_size __long

 int sys$expreg_64(
           struct _generic_64 *region_id_64,
           unsigned __int64 length_64,
           unsigned int acmode,
           unsigned int flags,
           void *(*(return_va_64)),
           unsigned __int64 *return_length_64);

#pragma __required_pointer_size __short
```

For more information about DEC C pointer size pragmas, see the *DEC C User's Guide for OpenVMS Systems*.

The following source code example shows the sys$expreg_64 function prototype referenced in a program.

```
#define __NEW_STARLET 1               /* Enable "New Starlet" features */

#include <starlet.h>                  /* Declare prototypes for system services */
#include <gen64def.h>                 /* Define GENERIC_64 type */
#include <vadef.h>                    /* Define VA$ constants */

#include <ints.h>                     /* Define 64-bit integer types */
#include <far_pointers.h>             /* Define 64-bit pointer types */

{
    int status;                       /* Ubiquitous VMS status value */
    GENERIC_64 region = { VA$C_P2 };  /* Expand in "default" P2 region */
    VOID_PQ p2_va;                    /* Returned VA in P2 space */
    uint64 length;                    /* Allocated size in bytes */
    extern uint64 page_size;          /* Page size in bytes */

    status = sys$expreg_64( &region, request_size, 0, 0, &p2_va, &length );
    ...

}
```

Table 4–5 lists the data structures that are used by the new function prototypes.

Table 4–5 Structures Used by _NEW_STARLET Prototypes

Structure Used by Prototype	Defined by Header File	Common Prefix for Structure Member Names	Description
struct _cluevthndl	cluevtdef.h	cluevthndl$	Cluster event handle
struct _fabdef	fabdef.h	fab$	File Access Block
struct _generic_64	gen64def.h	gen64$	Generic quadword structure
struct _ieee	ieeedef.h	ieee$	IEEE floating-point control structure
struct _ile2 [1]	iledef.h	ile2$	Item List Entry 2
struct _ile3 [1]	iledef.h	ile3$	Item List Entry 3
struct _iosa	iosadef.h	iosa$	I/O status area

[1]Use of this structure type is not required by the function prototypes in starlet.h. This structure type is provided as a convenience and can be used where it is appropriate.

(continued on next page)

Table 4–5 (Cont.) Structures Used by _NEW_STARLET Prototypes

Structure Used by Prototype	Defined by Header File	Common Prefix for Structure Member Names	Description
struct _iosb	iosbdef.h	iosb$	I/O Status Block
struct _lksb	lksbdef.h	lksb$	Lock Status Block
struct _rabdef	rabdef.h	rab$	RMS Record Access Block
struct _secid	seciddef.h	secid$	Global section identifier
struct _va_range	va_rangedef.h	va_range$	32-bit virtual address range ♦

4.14 Spiralog Version 1.0 (Alpha Only)

Alpha

The Spiralog file system, supported on OpenVMS Alpha Version 7.0, increases data write performance, ensures high availability, and provides rapid backup rates for VMSclusters and single computers. It is innovative technology, coupled with tight integration with Digital's PATHWORKS networking software and brings high-performance file services to diverse client systems.

The Spiralog file system is an option on OpenVMS Alpha and is fully compatible with the existing file system, Files–11. Both Spiralog and Files–11 volumes work simultaneously on the same system or within the same cluster. Data migration is simple: simply initialize a Spiralog volume and copy your Files–11 data to it. Your existing Files–11 applications run immediately without changes on the Spiralog file system.

Spiralog Version 1.0 is intended as an early developers kit for leading edge technology users. No separate license fee is required to run Spiralog; Spiralog license rights are included in the OpenVMS Alpha operating system license.

The Spiralog media will be available early in 1996 through the OpenVMS Software Products Library service or as a separate CD–ROM.

For more information about Spiralog, refer to the Spiralog documentation set. ♦

4.15 Dump File Compression Features (Alpha Only)

Alpha

The following dump file compression features are now available on the System Dump Analyzer utility (SDA).

4.15.1 Dump File Style

There are two types of dump files—a physical memory dump (also known as a full dump), and a dump of selected virtual addresses (also known as a selective dump). Both full and selective dumps may be produced in either compressed or uncompressed form. Compressed dumps save disk space and time taken writing the dump at the expense of a slight increase in time to access the dump with SDA. The SDA commands COPY/COMPRESS and COPY/DECOMPRESS can be used to convert an existing dump.

DUMPSTYLE, which specifies the method of writing system dumps, is a 32-bit mask. Table 4–6 shows how the bits are defined. Each bit can be set independently. The value of the SYSGEN parameter is the sum of the values of the bits that have been set. Remaining or undefined values are reserved to Digital.

Table 4–6 DUMPSTYLE Mask

Bit	Value	Description
0	0	0= Full dump (SYSGEN default). The entire contents of physical memory will be written to the dump file.
		1= Selective dump. The contents of memory will be written to the dump file selectively to maximize the usefulness of the dump file while conserving disk space.
1	2	0= Minimal console output.
		1= Full console output (includes stack dump, register contents, and so on.)
2	4	This bit is ignored on Alpha systems.
3	8	0= Do not compress.
		1= Compress.

In a physical memory dump, the DUMPSTYLE system parameter can be set to 0, 2, 8, or 10. Each value provides a full dump. The value of 0 yields an uncompressed dump with minimal console output; the value of 2 provides an uncompressed dump with full console output; the value of 8 provides a compressed dump with minimal console output; and the value of 10 provides a compressed dump with full console output. A physical memory dump requires that all physical memory be written to the dump file. This ensures the presence of all the page table pages required for SDA to emulate translation of system virtual addresses. These table pages include the level 1 page table of the current process, the shared level 2 page table that maps the system page table (SPT), and the level 3 page table pages that constitute the SPT.

In certain system configurations, it may be impossible to preserve the entire contents of memory in a disk file. For instance, a large memory system or a system with small disk capacity may not be able to supply enough disk space for a full memory dump. If the system dump file cannot accommodate all of memory, information essential to determining the cause of the system failure may be lost.

To preserve those portions of memory that contain information most useful in determining the causes of system failures, a system manager sets the value of the DUMPSTYLE system parameter to 1, 3, 9, or 11 to specify a dump of selected virtual address spaces. Each value provides a selective dump. The value of 1 yields an uncompressed dump with minimal console output; the value of 3 provides an uncompressed dump with full console output; the value of 9 provides a compressed dump with minimal console output; and the value of 11 provides a compressed dump with full console output. In a selective dump, related pages of virtual address space are written to the dump file as a unit called a logical memory block (LMB). For example, one LMB consists of the system and global page tables; another is the address space of a particular process. Those LMBs most likely to be useful in crash dump analysis are written first.

Table 4–7 compares full and selective style dump files.

Table 4–7 Comparison of Full and Selective Dump Files

Item	Full	Selective
Available Information	Complete contents of physical memory in use, stored in order of increasing physical address.	System page table, global page table, system space memory, and process and control regions (plus global pages) for all saved processes.
Unavailable Information	Contents of paged-out memory at the time of the system failure.	Contents of paged-out memory at the time of the system failure, process and control regions of unsaved processes, L1 page tables, and memory not mapped by a page table.
SDA Command Limitations	None.	The following commands are not useful for unsaved processes: SHOW PROCESS /CHANNELS, SHOW PROCESS/IMAGE, SHOW PROCESS/RMS, SHOW STACK, and SHOW SUMMARY/IMAGE.

4.15.1.1 Controlling the Size of Page Files and Dump Files

You can adjust the size of the system page file and dump file using AUTOGEN (the recommended method) or by using SYSGEN.

AUTOGEN automatically calculates the appropriate sizes for page and dump files. AUTOGEN invokes the System Generation utility (SYSGEN) to create or change the files. However, you can control sizes calculated by AUTOGEN by defining symbols in the MODPARAMS.DAT file. The file sizes specified in MODPARAMS.DAT are copied into the PARAMS.DAT file during AUTOGEN's GETDATA phase. AUTOGEN then makes appropriate adjustments in its calculations.

Although Digital recommends using AUTOGEN to create and modify page and dump file sizes, you can use SYSGEN to directly create and change the sizes of those files.

The sections that follow discuss how you can calculate the size of a dump file.

See the Bookreader version of the *OpenVMS System Manager's Manual* for detailed information about using AUTOGEN and SYSGEN to create and modify page and dump file sizes.

4.15.1.2 Writing to the System Dump File

OpenVMS Alpha writes the contents of the error-log buffers, processor registers, and memory into the system dump file, overwriting its previous contents. If the system dump file is too small, OpenVMS Alpha cannot copy all memory to the file when a system failure occurs.

SYS$SYSTEM:SYSDUMP.DMP (SYS$SPECIFIC:[SYSEXE]SYSDUMP.DMP) is furnished as an empty file in the OpenVMS Alpha software distribution kit. To successfully store a crash dump, SYS$SYSTEM:SYSDUMP.DMP must be enlarged to hold all of the page tables required for SDA to emulate system virtual address translation.

To calculate the correct size for a physical memory dump to SYS$SYSTEM:SYSDUMP.DMP, use the following formula:

```
size-in-blocks(SYS$SYSTEM:SYSDUMP.DMP)
        = size-in-pages(physical-memory) * blocks-per-page
        + number-of-error-log-buffers * blocks-per-buffer
        + size-in-pages(physical-memory)/512
        + 2
```

Use the DCL command SHOW MEMORY to determine the total size of physical memory on your system. There is a variable number of error log buffers in any given system, depending on the setting of the ERRORLOGBUFFERS system parameter. The size of each buffer depends on the setting of the ERLBUFFERPAGES parameter. (See the Bookreader version of the *OpenVMS System Manager's Manual* for additional information about these parameters.)

To calculate the correct size for a compressed physical dump to SYS$SYSTEM:SYSDUMP.DMP, use two-thirds of the size calculated for an uncompressed dump.

Digital recommends using AUTOGEN to create and modify dump sizes for selective dumps. For a compressed selective dump, AUTOGEN will size the file at two-thirds the size on an uncompressed dump.

4.15.1.3 Writing to the System Page File

If SYS$SYSTEM:SYSDUMP.DMP does not exist, the operating system writes the dump of physical memory into SYS$SYSTEM:PAGEFILE.SYS, the primary system page file, overwriting the contents of that file.

If the SAVEDUMP system parameter is set, the dump file is retained in PAGEFILE.SYS when the system is booted after a system failure. If the SAVEDUMP parameter is not set (clear), which is the default, OpenVMS Alpha uses the entire page file for paging and any dump written to the page file is lost. (To examine or change the value of the SAVEDUMP parameter, consult the *OpenVMS System Manager's Manual*.)

To calculate the minimum size for a physical memory dump to SYS$SYSTEM:PAGEFILE.SYS, use the following formula:

```
size-in-blocks(SYS$SYSTEM:PAGEFILE.SYS)
        = size-in-pages(physical-memory) * blocks-per-page
        + number-of-error-log-buffers  * blocks-per-buffer
        + size-in-pages(physical_memory)/512
        + 2
        + value of the system parameter RSRVPAGCNT
```

Note that this formula calculates the minimum size requirement for saving a physical dump in the system's page file. Digital recommends that the page file be a bit larger than this minimum to avoid hanging the system. Also note that you can only write the dump of physical memory into the primary page file (SYS$SYSTEM:PAGEFILE.SYS). Secondary page files cannot be used to save dump file information.

To calculate the correct size for a compressed physical dump to SYS$SYSTEM:PAGEFILE.SYS, use two-thirds of the size calculated for an uncompressed dump before system parameter RSRVPAGCNT is added in, and then add in the value of RSRVPAGCNT.

It is not recommended to use a selective dump (DUMPSTYLE=1) style with PAGEFILE.SYS. If the PAGEFILE.SYS is used for a selective dump, and if the PAGEFILE.SYS is not large enough to contain all the logical memory blocks, the dump fills the entire page file and the system may hang on reboot. When selective dumping is set up, all available space is used to write out the logical memory blocks. If the page file is large enough to contain all of physical memory, there is no reason to use selective dumping. A full memory dump (DUMPSTYLE=0) should be used.

Writing crash dumps to SYS$SYSTEM:PAGEFILE.SYS presumes that you will later free the space occupied by the dump for use by the pager. Otherwise, your system may hang during the startup procedure. To free this space, you can do one of the following:

- Include SDA commands that free dump space in the site-specific startup command procedure.)
- Use the SDA COPY command to copy the dump from SYS$SYSTEM:PAGEFILE.SYS to another file. Use the SDA COPY command instead of the DCL COPY command because the SDA COPY command causes the pages occupied by the dump to be freed from the system's page file.
- If you do not need to copy the dump elsewhere, issue an ANALYZE /CRASH_DUMP/RELEASE command. When you issue this command, SDA immediately releases the pages to be used for system paging, effectively deleting the dump. Note that this command does not allow you to analyze the dump before deleting it. ♦

4.16 New SMBMSG$V_NO_INITIAL_FF Symbol for SMBMSG$K_PRINT_CONTROL Message Item Code

The SMB$READ_MESSAGE_ITEM routine of the Symbiont/Job Controller Interface (SMB) has a new symbol, SMBMSG$V_NO_INITIAL_FF, for the SMBMSG$K_PRINT_CONTROL message item code.

Table 4–8 defines the SMBMSG$V_NO_INITIAL_FF symbol.

Table 4–8 SMBMSG$V_NO_INITIAL_FF Symbol

Symbol	Description
SMBMSG$V_NO_INITIAL_FF	The symbiont suppresses the initial formfeed if this bit is turned ON.

For more information, see the Bookreader version of the *OpenVMS Utility Routines Manual*.

4.17 System Services

The following sections describe new system services features.

4.17.1 Fast IO System Services (Alpha Only)

Alpha

Fast IO refers to a suite of new system services that provide an alternative to the $QIO system service.

For more information about Fast IO, see Chapter 5. ♦

4.17.2 New QIO Attribute, ATR$C_FILE_SYSTEM_INFO

Use the new QIO attribute, ATR$C_FILE_SYSTEM_INFO, to check which file system created a file or directory.

The attribute is read-only and can have the following values:

High Byte	Low Byte	File System
1	1	Files–11 A (ODS-1)[1]
2	1	Files–11 B (ODS-2)
0	0	Files–11 C (ISO9660) or Files–11 D (High Sierra)

[1]VAX only

4.17.3 New $CREPRC Argument

The new **node** argument for the $CREPRC system service allows an application to specify that a detached process is to be created on another VMScluster node. The argument is the address of a character string descriptor pointing to a 1- to 6-character SCS node name. For detailed information about creating a process on another node, see the OpenVMS System Services manuals.

4.17.4 New System Services to Support CPU Scheduling (Alpha Only)

Alpha

OpenVMS Version 7.0 contains new system services to allow you to do the following:

- Create and modify a set of user-defined process capabilities
- Create and modify a set of user-defined CPU capabilities to match those in the process
- Allow a process to share affinity with a subset of the active CPU set in an SMP configuration

Service	Description
$CPU_CAPABILITIES	Allows modification of the user capability set for a specified CPU, or for the global CPU default.
$PROCESS_CAPABILITIES	Allows modification of the user capability set for a specified process thread, or for the global process default.
$PROCESS_AFFINITY	Allows modification of the CPU affinity set for a specified process thread.
$SET_IMPLICIT_AFFINITY	Controls or retrieves the activation state of the implicit affinity capability for a specified process thread, or for the global process default.

For detailed information about these services, see the OpenVMS System Services manuals.♦

4.18 CPU Scheduling (Alpha Only)

Alpha

The algorithms used by OpenVMS Alpha scheduling to select the next active Kernel thread follow symmetric rules in a multiprocessing configuration. Each processor is responsible for using the standard priority and preemption rules to determine whether its active process thread requires a state change, and, with minor exceptions, each processor behaves as if it is alone in a single-processor configuration. This behavior has benefits in predictability and consistency—particularly in the real-time priority ranges.

However, application performance on the Alpha architecture is sensitive to the existence of the Kernel thread's context in the processor's cache and translation buffer (TB). Maximizing this context by binding a running thread to specific processor often shows a throughput improvement that can outweigh the benefits of the symmetric scheduling model. Particularly in larger CPU configurations and higher-performance server applications, the ability to control the distribution of Kernel threads throughout the active CPU set has become increasingly important.

The OpenVMS scheduling mechanisms have always required some CPU scheduling functions for its internal operations. To provide this functionality to the application level, a new set of features have been added to the system service and command interfaces. These changes have been made in three areas:

- Capabilities

  Specify a set of "resources" that a CPU in the active set must have defined before it is allowed to contend for a Kernel thread's execution.

- Explicit Affinity

  Allow a Kernel thread to specify an exact set of CPUs on which it can execute.

- Implicit Affinity

  Allow a Kernel thread to participate in a relaxed scheduling mode where processor biasing and load balancing are employed to maximize its cache and TB context.

4.18.1 Capabilities

Capabilities can be thought of as resources assigned to CPUs that a process thread needs to execute correctly. There are currently four system capabilities used to control system states or functions:

- PRIMARY - requires that the process run on the primary CPU. This is used primarily for I/O and timekeeping functions. Because the functions of the primary capability theoretically could migrate from CPU to CPU in the configuration, this capability is owned by only one of them at a time. The Kernel thread requiring this capability is only allowed to run on the processor that has it at the time.

- RUN - controls the ability of a CPU to execute any process at all. Every process requires this attribute and if the CPU does not have it, scheduling for that CPU comes to a halt in a recognized state. STOP/CPU uses this capability when it is trying to quiesce the CPU and remove it from the active set.

- QUORUM - used in a clustered environment when another node wants this one to come to a quiescent state until a cluster event has complete. Like the RUN capability, this is a required attribute for every Kernel thread and CPU for scheduling to occur.
- VECTOR - used to indicate that a vector processor is associated with this CPU. This is obsolete on Alpha systems, but is retained as a compatibility feature with OpenVMS VAX.

4.18.1.1 User Capabilities

OpenVMS Alpha has added new support for 16 user capabilities. This provides the same type of resource control as system capabilities, but the definition of the individual user capabilities are left up to external consumers.

Unlike the static definitions of system capabilities, user capabilities will have meaning only in the context of the Kernel threads that define them. Through system service interfaces and DCL commands, an execution thread will be able to set specific bits in the capability masks of a CPU to give it a "resource", and to set specific bits in the Kernel thread capability masks to require that resource as an execution criteria.

Both the system and user capability types will affect the OpenVMS scheduling mechanisms in the same way, but only the user set can be changed through the new user interfaces. Access to system capabilities will continue to be through the defined privileged internal interfaces.

The assignment of a user capability to a CPU has no direct effect on the scheduling dynamics of the system; it only indicates that the specified CPU is capable of handling any Kernel thread that requires that particular resource. If a Kernel thread does not require that resource, the scheduling mechanism ignores the CPU's additional capability and schedules the thread based on other criteria.

Assigning a user capability to a specific Kernel thread does have an impact on the scheduling state of that entity. For the Kernel thread to be scheduled on a CPU in the active set, that CPU must have the capability assigned prior to the scheduling attempt. If no CPU currently has the correct set of capability requirements, the process is placed into a wait state until a CPU becomes available with the right configuration. As with system capabilities, user process capabilities are additive; for a CPU to schedule the process it must have the full complement of required capabilities.

4.18.1.2 Scope of User Capabilities

User capabilities, once given to a CPU, do not change until specifically modified by another system service or until a reboot. These values will be retained even across a processor shutdown and restart sequence using the STOP/CPU and START/CPU command.

Process user capabilities, however, survive in two forms over the life of the Kernel thread. The thread maintains a permanent capability mask that reinitializes the current capability mask value at every image rundown. The current values exist only as long as the image is active; changes to them disappear along with the image unless the modified capabilities are changed in the permanent mask as well.

All user interfaces to the process user capability features allow either current or permanent masks to be modified; however, changes to the permanent mask are automatically applied to the current mask as well.

4.18.1.3 Capability System Services

Access to the process and CPU user capability features from an application is through a new set of system services:

- $CPU_CAPABILITIES
- $PROCESS_CAPABILITIES

For a complete description of the service call interfaces, see the appropriate sections in the *OpenVMS System Services Reference Manual: A–GETMSG* and *OpenVMS System Services Reference Manual: GETQUI–Z.*

An additional feature of these services is that they can be used to modify the system global defaults for both processor activation and image initialization.

As a CPU is booted or brought into the active set for the first time, the capability values placed in its database are taken from default values in the global cell SCH$GL_DEFAULT_CPU_CAP. The $CPU_CAPABILITIES service provides a way to modify the default cell, instead of a specific CPU, which affects any processor initialization that follows the service call.

Permanent Kernel thread user capabilities are initialized at process creation from the values in a global cell SCH$GL_DEFAULT_PROCESS_CAP. The permanent mask is then used to initialize the current capabilities mask on the first image activation and every subsequent rundown. The $PROCESS_CAPABILITIES service provides a means to modify the default cell instead of a specific Kernel thread, which affects any process that is created following the service call.

4.18.1.4 /CAPABILITIES Qualifier

SET PROCESS /[NO]CAPABILITIES[/SET=(n)][/CLEAR=(n)][/PERMANENT]

The /CAPABILITIES command qualifier allows bits in the user process capability mask to be set or cleared individually, in groups, or all at once. This qualifier is mutually exclusive with the /AFFINITY qualifier.

The /NOCAPABILITIES qualifier clears all user capability bits currently set, based on the setting of the /PERMANENT qualifier. Specifying /CAPABILITIES itself has no direct effect other than to indicate the target of the operations specified by the following secondary qualifiers:

/SET=(n,[,...])	Sets all user capabilities defined by the position values n, where n has the range of 1 to 16.
/CLEAR=(n,[,...])	Clears all user capabilities defined by the position values n, where n has the range of 1 to 16.
/PERMANENT	Forces the operations to be performed on the permanent user mask as well as the current, effectively making the changes permanent for the life of the thread or process. (The default behavior is to affect only the running image copy of the capabilities.)

The secondary qualifiers can all be used at once as long as the set of user capability bits defined in the /SET and /CLEAR parameters does not overlap.

The privileges required to execute this command match those required by the $PROCESS_AFFINITY system service. ALTPRI is the base privilege required to make any modifications, and the only privilege required to modify the current owner's process/thread. To make modifications in the same UIC group, GROUP is required. Otherwise, to make modifications to any unrelated process/thread, WORLD privilege is required.

As with the other SET PROCESS command qualifiers, the bit operations occur on the current process if no /IDENTIFICATION qualifier or explicit process name parameter is specified. Note that the /IDENTIFICATION qualifier allows this command to affect individual Kernel Thread PIDs; since each KTB is a separate runnable entity, these commands treat them as discrete entities in terms of capabilities. Specifying a process name does not imply that all threads associated with the process are affected; the SET PROCESS command affects only the initial thread of a multithreaded process.

4.18.2 Explicit Affinity

While capabilities and explicit affinity overlap in their functional behavior, they are two discrete scheduling mechanisms. Explicit affinity - subsetting the number of CPUs on which a process can execute - has precedence over capability and provides an explicit binding operation between a Kernel thread and a CPU. Explicit affinity forces the OpenVMS scheduling mechanisms to consider only the CPU set it allows, and then applies the capability scheduling criteria to see if they are appropriate.

OpenVMS has always allowed explicit affinity for a specific Kernel thread, but until now the mechanism for manipulating the feature was through an internal, privileged interface. The ability to restrict the availability of the active CPU set through user system service and DCL command interfaces is a new feature with this release.

Explicit affinity is not in effect at Kernel thread creation; standard OpenVMS symmetric scheduling rules apply to select the next processor on which the thread will execute. However, once explicit affinity is enabled through the user interfaces, it is in effect until all bindings are manually removed, automatically disabling the feature and reasserting the default scheduling rules. While enabled, the explicit affinity settings for a thread can be modified to include any or all of the active processors currently in the SMP system configuration.

Like user capability, explicit affinity has performance benefits from maximizing a Kernel thread's cache and TB context on specific processors. Instead of dealing with user-defined generic resources, however, explicit affinity improves performance by directly controlling the underlying configuration layout. Although both mechanisms achieve the same functional effect, explicit affinity is applied before user capability in the scheduling critera; for some applications this may provide a greater degree of control.

4.18.2.1 Scope of Explicit Affinity

Explicit affinity survives in two forms over the life of a Kernel thread. In the same manner as capabilities, the thread maintains a permanent mask that reinitializes the current affinity mask at every image rundown. The current bindings exist only as long as the image is active; changes to the current mask disappear along with the image unless the bindings are made in the permanent mask as well.

All user interfaces to the explicit affinity features allow either current or permanent masks to be modified; however, changes to the permanent mask are automatically applied to the current mask as well.

4.18.2.2 Explicit Affinity System Service

Application control of explicit affinity is through the new system service:

$PROCESS_AFFINITY

For a complete description of the service call interface, see the appropriate section in the *OpenVMS System Services Reference Manual: GETQUI–Z.*

4.18.2.3 /AFFINITY Qualifier

The /AFFINITY qualifier allows bits in the affinity mask to be set or cleared individually, in groups, or all at once. This qualifier is mutually exclusive with the /CAPABILITIES qualifier.

The /NOAFFINITY qualifier clears all affinity bits currently set, based on the setting of the /PERMANENT qualifier. Specifying /AFFINITY itself has no direct effect other than to indicate the target of the operations specified by the following secondary parameters:

/SET=(n,[,...])	Sets all CPU affinities defined by the position n, where n has a range of 1 to 32 and is restricted to the set of currently active CPUs.
/CLEAR=(n,[,...])	Clears all CPU affinities defined by the position values n, where n has a range of 1 to 32 and is restricted to the set of currently active CPUs.
/PERMANENT	Forces the operations to be performed on the permanent user mask as well as the current, effectively making the changes permanent for the life of the thread or process. (The default behavior is to affect only the running image copy of the affinities.)

The secondary qualifiers can all be used at once as long as the set of affinity bits defined in the /SET and /CLEAR parameters do not overlap.

The privileges required to execute this command match those required by the $PROCESS_AFFINITY system service. ALTPRI is the base privilege required to make any modifications, and the only privilege required to modify the current owner's process/thread. To make modifications in the same UIC group, GROUP is required. Otherwise, to make modifications to any unrelated process/thread, WORLD privilege is required.

As with the other SET PROCESS command qualifiers, the bit operations occur on the current process if no /IDENTIFICATION qualifier or explicit process name parameter is specified. Note that the /IDENTIFICATION qualifier allows this command to affect individual Kernel Thread PIDs; since each KTB is a separate runnable entity, these commands treat them as discrete entities in terms of affinities. Specifying a process name does not imply that all threads associated with the process are affected; the SET PROCESS command affects only the initial thread of a multithreaded process.

4.18.3 Implicit Affinity

Implicit affinity is a third mechanism that can be used by a specific process thread to improve its performance. Unlike explicit affinity and user capabilities, implicit affinity is a system-directed load balancing mechanism used to maximize the thread's cache and TB context.

With implicit affinity enabled, a Kernel thread is subject to relaxed scheduling rules in determining on which available processor it will next execute. Instead of immediately scheduling on another CPU that fits the normal priority and preemption rules, the thread may be biased towards the last CPU on which it was active.

The assumption in this feature is that there is enough cache context remaining from the previous association to outweigh the benefit of an immediate scheduling switch to another processor. On the Alpha architecture it has been shown that maintaining cache and TB context has a significant potential for performance improvement on both the Kernel thread and system level. If sufficient context is still available on the previous CPU, the tradeoff with the scheduling delay can be very worthwhile.

However, because this concept contradicts the OpenVMS scheduling algorithms in their most literal sense, implicit affinity cannot be a system default. Care must be taken to identify which Kernel threads can benefit most from the relaxed scheduling and dynamic load balancing rules.

Implicit affinity fits into the lower end of the other CPU scheduling mechanisms; both explicit affinity and user capabilities take precedence in determining which CPUs are available for implicit affinity selection. The actual scheduling rules are no different with the other two mechanisms in effect, but the set of available CPUs to choose from may be restricted by them.

The only access to the implicit affinity feature is through the new system service:

$SET_IMPLICIT_AFFINITY

This service provides an interface to enable or disable implicit affinity for a specific Kernel thread, or, through the global system default, for all newly-created processes. The service also provides a means to read the current state of the feature for a specific Kernel thread.

For a complete description of the service call interface, see the appropriate section in the *OpenVMS System Services Reference Manual: GETQUI–Z.*

4.18.4 Informational Services

Other changes have been made to the DCL command set and the informational system services, providing additional configuration and control context to the user and application interfaces. The following sections describe features that are complementary to the primary CPU Scheduling functions.

4.18.4.1 DCL SHOW CPU

The SHOW CPU command has been changed to display thread bindings due to explicit affinity. It will also display threads with user capabilities that have differing values from the default process creation value.

4.18.4.2 SDA SHOW PROCESS

The SDA display for SHOW PROCESS has been changed to display the user capability and permanent explicit affinity masks for all threads in the process. The required capabilities header has been eliminated from the first page display and the additional context displayed on each thread's display page.

4.18.4.3 $GETSYI - General System Information

The following item codes listed in Table 4–9 were added to the $GETSYI system service to return configuration information in an SMP environment.

Table 4–9 $GETSYI item codes for CPU Scheduling

Item code	Description
SYI$_ACTIVE_CPU_MASK	$GETSYI returns a mask of the CPUs actively participating in the current boot of the SMP system. The service returns this information for the local node only.
SYI$_AVAIL_CPU_MASK	$GETSYI returns a mask of the present and available CPUs participating in the current boot of the SMP system. The service returns this information for the local node only.
SYI$_PRIMARY_CPUID	$GETSYI returns the ID of the primary CPU in the current boot of the SMP system. The service returns this information for the local node only.
SYI$_MAX_CPUS	$GETSYI returns the maximum number of CPUs that can exist in the current configuration of the SMP system. The service returns this information for the local node only.
SYI$_CPUCAP_MASK	$GETSYI returns an array of quadword user capability masks for all CPUs in the system. This array is indexed by CPU ID and contains as many elements as is space specified by the buffer length field in the item descriptor. To minimize wasted space, a prior call to $GETSYI with SYI$_MAX_CPUS will provide the number of CPUs that need to be retrieved. Multiplying that value by 8 bytes for each quadword will provide the value to be written in the buffer length field of the item descriptor. The service returns this information for the local node only.

4.18.4.4 $GETJPI - Kernel Thread Information

The following item codes listed in Table 4–10 are added to return information about user capabilities and explicit affinity for a specified Kernel thread in an SMP configuration.

Table 4–10 $GETJPI item codes for CPU Scheduling

Item code	Description
JPI$_CURRENT_USERCAP_MASK	$GETJPI returns the current user capability mask for the associated Kernel thread.
JPI$_PERMANENT_USERCAP_MASK	$GETJPI returns the permanent user capability mask for the associated Kernel thread.
JPI$_CURRENT_AFFINITY_MASK	$GETJPI returns the current explicit affinity mask for the associated Kernel thread.
JPI$_PERMANENT_AFFINITY_MASK	$GETJPI returns the permanent explicit affinity mask for the associated Kernel thread.♦

4.19 Alternative to Local Event Flags

With OpenVMS Version 7.0, event flags are divided into five clusters. The new special local cluster 4 supports only EFN 128. The EFN 128, symbolically, EFN$C_ENF, is intended for use with the wait forms of services, such as SYS$QIOW or SYS$ENQW, or SYS$SYNCH system service. ENF$C_ENF does not need to be initialized, nor does it need to be reserved or freed. Multiple threads of execution may concurrently use EFN$C_ENF without interference.

If EFN$C_ENF is used with system services, such as SYS$SETEF, SYS$READ, and SYS$CLREF, it performs as if always set. EFN$C_ENF can be used to eliminate the chance for event flag overlap. It can also be used when you don't care about the event flag, for example, using SYS$QIO with an AST completion.

For more information, see the Bookreader version of the *OpenVMS Programming Concepts Manual*.

4.20 Run-Time Library (RTL) Routines

This section describes new features for the run-time library (RTL) routines.

4.20.1 Using LIB$CREATE_DIR to Create Large Directories

The LIB$CREATE_DIR routine has a new optional argument, **initial-allocation**, that you can use to specify the initial number of blocks to be allocated to the directory.

This argument is useful for creating large directories, for example MAIL.DIR;1. It can improve performance by avoiding the need for later dynamic expansion of the directory.

See the *OpenVMS RTL Library (LIB$) Manual* for more information.

4.21 Wind/U Version 3.0 Run Time on OpenVMS Systems

For this release, the OpenVMS Version 7.0 CD–ROM distribution kit will include Wind/U Version 3.0 run-time binaries.

Wind/U is a product that supports the Win32 API and enables Windows applications to run on OpenVMS platforms. It is a library of callable routines that translate Win32 API calls into the appropriate OpenVMS system services.

To run Windows applications under OpenVMS, users need to recompile and relink their applications. Wind/U Version 3.0 provides tight integration with Visual C++ including Microsoft Foundation Class 4 and supports other Windows NT and Windows 95 features, such as the Component Object Model (COM), Object Linking and Embedding (OLE), Visual Editing, OLE Automation, and Drag and Drop.

For licensing information and additional information on the developer's kit, contact Bristol Technology at:

241 Ethan Allen Highway, Ridgefield, CT 06877 USA
203 438-6969
email:info@bristol.com
http://www.bristol.com

4.22 ZIC Utility

The zic command allows the zic compiler to create binary time zone conversion information files from a time zone source file.

4.22.1 Format

zic [–v] ["–L" leapseconds] [–d directory] [–y yearistype] [filename]

4.22.1.1 Parameters

–v

Flags if a year that appears in a data file is outside the range of years representable by time values.

"–L"

Reads leap second information from the file with the given name. If this option is not used, no leap second information appears in the output files.

–d

Creates time conversion information files in the named directory rather than in the standard directory.

–y

Uses the given command file rather than yearistype when checking year types.

filename

Source file from which zic reads its input.

4.22.1.2 Qualifiers

None

4.22.2 Description

The zic command allows the zic compiler to read text from the file(s) named on the command line, and then creates the time conversion information files specified with this input. If a file name is –, then the standard input is read.

Input lines consist of fields. Any number of white space characters separate the fields from each other. Leading and trailing white spaces on input lines are ignored. An unquoted number sign **#**, the sharp character, in the input line introduces a comment that extends to the end of the line where this sign appears. White space characters and sharp characters can be enclosed in double marks, **"** **"**, if they are to be used as part of a field. Any line that is blank after comment stripping is ignored. Non-blank lines are expected to be one of three types:

- rule lines
- zone lines
- link lines

4.22.2.1 Rule Lines

A rule line has the following form:

```
Rule  NAME  FROM    TO TYPE IN ON      AT   SAVE LETTER/S
```

An example is as follows:

```
Rule  USA   1969 1973 - Apr lastSun    2:00 1:00 D
```

The rule line consists of the following fields:

NAME

Gives the arbitrary name of the set of rules that this rule is part of.

FROM

Gives the first year in which the rule applies. The word minimum, or an abbreviation, means the minimum year with a representable time value. The word maximum, or an abbreviation, means the maximum year with a representable time value.

TO

Gives the final year in which the rule applies. In addition to minimum and maximum as defined in FROM, minimum or maximum (or an abbreviation) only may be used to repeat the value of the FROM field.

TYPE

Gives the type of year in which the rule applies. If TYPE is a −, then the rule applies in all years between FROM and TO inclusively. Zic executes the following command to check the type of year:

```
yearistype year type
```

An exit status of 1 means that the year is of the given type; an exit status of 5 means that the year is not of the given type.

Gives the month in which the rule takes effect. Month names may be abbreviated.

ON

Gives the day on which the rule takes effect. Table 4–11 shows the recognized forms.

Table 4–11 Day the Rule Becomes Effective

Form	Meaning
5	The fifth of the month
lastSun	The last Sunday in the month
lastMon	The last Monday in the month
Sun>=8	First Sunday on or after the eighth
Sun<=25	Last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note that there must be no spaces within the ON field.

AT

Gives the time of day on which the rule takes effect. Table 4–12 shows the recognized forms:

Table 4–12 Time of Day the Rule Becomes Effective

Form	Meaning
2	Time in hours
2:00	Time in hours and minutes
15:00	24-hour format time (for times after noon)
1:28:14	Time in hours, minutes, and seconds

Any of these forms may be followed by the letter **w** if the given time is local *wall clock* time, or the letter **s** if the time is local *standard* time. In the absence of either the letter **w**, or the letter **s**, *wall clock* time is assumed.

SAVE

Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the AT field, although, of course, the letter **w** and **w** suffixes are not used.

LETTER/S

Gives the *variable part* of time zone abbreviations to be used when this rule is in effect; as for example, the *S* or *D* in *EST* or *EDT*. If this field is −, the variable part is null.

4.22.2.2 Zone Lines

A zone line has the following form:

```
 "Zone NAME              GMTOFF  RULES/SAVE FORMAT UNTIL]"
```

An example is as follows:

```
Zone Australia/South-west 9:30   Aus         CST  1987 Mar 15 2:00
```

The zone line consists of the following fields:

NAME

Gives the name of the time zone. This name is used in creating the time conversion information file for the zone.

GMTOFF

Gives the amount of time to add to GMT to get standard time in this zone. This field has the same format as the AT and SAVE fields of rule lines. If time must be subtracted from GMT, begin the field with a minus sign.

RULES/SAVE

Gives the name of the rule(s) that apply in the time zone, or alternately, an amount of time to add to local standard time. If this field is −, standard time always applies in the time zone.

FORMAT

Gives the format for time zone abbreviations in this time zone. The pair of characters **%s** is used to show where the variable part of the time zone abbreviation goes.

UNTIL

Gives the time at which the GMT offset, or the rule(s) change for a location. It is specified as the following:

A year
A month
A day
A time of day

If UNTIL is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a *continuation* line. The continuation line has the same form as the zone line except that the string *Zone* and the name are omitted, for the continuation line places information starting at the time specified in the UNTIL field in the previous line in the file used by the previous line.

Continuation lines may contain an UNTIL field, just as zone lines do, indicating that the next line is a further continuation.

4.22.2.3 Link Lines

A link line has the following form:

```
"Link      LINK-FROM      LINK-TO"
```

An example is as follows:

```
Link       US/Eastern     EST5EDT
```

In the OpenVMS implementation, Link is interpreted as a copy. Thus the above line copies the information from US/Eastern to EST5EDT.

The LINK-FROM field should appear as the NAME field in some zone line. The LINK-TO field is used as an alternate name for that zone.

Except for continuation lines, lines may appear in any order in the input.

Note

For areas with more than two types of local time, you may need to use local standard time in the AT field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

4.22.3 Example

The following is a zic command line example:

```
$ zic -v  "-L" leapseco -d [-] myafrica
```

This example causes zic to compile the time zone source file myafrica. Based on the parameters selected it:

1. Flags years outside the representable range
2. Builds an output file with leapsecond corrections applied
3. Puts the result in the current directory

For more information about date/time functions, see the *DEC C Run-Time Library Reference Manual for OpenVMS Systems*.

5
Optional Features for Improving I/O Performance

Alpha

OpenVMS Alpha Version 7.0 includes two new features for providing dramatically improved I/O performance: Fast I/O and Fast Path. These features are designed to promote OpenVMS as a leading platform for database systems. Performance improvement results from reducing the CPU cost per I/O request and improving SMP scaling of I/O operations. The CPU cost per I/O is reduced by optimizing code for high-volume I/O and by using better SMP CPU memory cache. SMP scaling of I/O is increased by reducing the number of spinlocks taken per I/O and by substituting finer-granularity spinlocks for global spinlocks.

The improvements follow a natural division that already exists between the device-independent and device-dependent layers in the OpenVMS I/O subsystem. The device-independent overhead is addressed by Fast I/O, which is a set of lean system services that can substitute for certain $QIO operations. Using these services requires some coding changes in existing applications, but the changes are usually modest and well-contained. The device-dependent overhead is addressed by Fast Path, which is an optional performance feature that creates a ‘‘fast path’’ to the device. It requires no application changes.

Fast I/O and Fast Path can be used independently. However, together they can provide on the order of a 45% reduction in CPU cost per I/O on uniprocessor systems and a 52% reduction on multiprocessor systems.

5.1 Fast I/O

Fast I/O is a set of three new system services that were developed as a $QIO alternate built for speed. These services are not a $QIO replacement; $QIO is unchanged, and $QIO interoperation with these new services is fully supported. Rather, the new services substitute for a subset of $QIO operations, namely, only the high-volume read/write I/O requests.

The Fast I/O services support 64-bit addresses for data transfers to and from disk and tape devices.

While Fast I/O services are available on OpenVMS VAX, the performance advantage applies only to OpenVMS Alpha. OpenVMS VAX has an RTL compatibility package that translates the new Fast I/O service requests to $QIO system service requests, so that one set of source code can be used on both VAX and Alpha systems.

Optional Features for Improving I/O Performance
5.1 Fast I/O

5.1.1 Fast I/O Benefits

The performance benefits of Fast I/O result from streamlining high-volume I/O requests. The Fast I/O system service interfaces are optimized to avoid the overhead of general-purpose services. For example, IRPs are now permanently allocated and used repeatedly for I/O rather than allocated and deallocated anew for each I/O.

The greatest benefits stem from having user data buffers and user I/O status structures permanently locked down and mapped using system space. This allows Fast I/O to do the following:

- For direct I/O, avoid per-I/O buffer lockdown or unlocking.
- For buffered I/O, avoid allocation and deallocation of a separate system buffer, since the user buffer is always addressable.
- Complete Fast I/O operations at IPL 8, thereby avoiding the interrupt chaining usually required by the more general-purpose $QIO system service. For each I/O, this eliminates the IPL 4 IOPOST interrupt and a kernel AST.

In total, Fast I/O services eliminate four spinlock acquisitions per I/O (two for the MMG spinlock and two for the SCHED spinlock). The reduction in CPU cost per I/O is 20% for uniprocessor systems and 10% for multiprocessor systems.

5.1.2 Using Buffer Objects

The lockdown of user-process data structures is accomplished by buffer objects. A ‘‘buffer object’’ is process memory whose physical pages have been locked in memory and double-mapped into system space. After creating a buffer object, the process remains fully pageable and swappable and the process retains normal virtual memory access to its pages in the buffer object.

If the buffer object contains process data structures to be passed to an OpenVMS system service, then the OpenVMS system can use the buffer object to avoid any probing, lockdown, and unlocking overhead associated with these process data structures. Additionally, double-mapping into system space allows the OpenVMS system direct access to the process memory from system context.

To date, only the $QIO system service and the new Fast I/O services have been changed to accept buffer objects. For example, a buffer object allows a programmer to eliminate I/O memory management overhead. On each I/O, each page of a user data buffer is probed and then locked down on I/O initiation and unlocked on I/O completion. Instead of incurring this overhead for each I/O, it can be done once at buffer object creation time. Subsequent I/O operations involving the buffer object can completely avoid this memory management overhead.

Two system services can be used to create and delete buffer objects, respectively, and can be called from any access mode. To create a buffer object, the $CREATE_BUFOBJ system service is called. This service expects as inputs an existing process memory range and returns a buffer handle for the buffer object. The buffer handle is an opaque identifier used to identify the buffer object on future I/O requests. The $DELETE_BUFOBJ system service is used to delete the buffer object and accepts as input the buffer handle. Although image rundown deletes all existing buffer objects, it is good form for the application to clean up properly.

A new 64-bit equivalent version of the $CREATE_BUFOBJ system service ($CREATE_BUFOBJ_64) can be used to create buffer objects from the new 64-bit P2 or S2 regions. The $DELETE_BUFOBJ system service can be used to delete 32-bit or 64-bit buffer objects.

Buffer objects require system management. Because buffer objects tie up physical memory, extensive use of buffer objects require system management planning. All the bytes of memory in the buffer object are deducted from a new systemwide SYSGEN parameter called MAXBOBMEM (maximum buffer object memory). System managers must set this parameter correctly for the application loads that run on their systems.

The MAXBOBMEM parameter defaults to 100 Alpha pages, but for applications with large buffer pools it will likely be set much larger. To prevent user-mode code from tying up excessive physical memory, user-mode callers of $CREATE_BUFOBJ must have a new system identifier, VMS$BUFFER_OBJECT_USER, assigned. This new identifier is automatically created in an OpenVMS Version 7.0 upgrade if the file SYS$SYSTEM:RIGHTSLIST.DAT is present. The system manager can assign this identifier with the DCL command SET ACL command to a protected subsystem or application that creates buffer objects from user mode. It may also be appropriate to grant the identifier to a particular user with the Authorize utility command GRANT/IDENTIFIER (for example, to a programmer who is working on a development system).

There is currently a restriction on the type of process memory that can be used for buffer objects. Global section memory cannot be made into a buffer object.

5.1.3 Differences Between Fast I/O Services and $QIO

The precise definition of high-volume I/O operations optimized by Fast I/O services is important. I/O that does not comply with this definition either is not possible with the Fast I/O services or is not optimized. The characteristics of the high-volume I/O optimized by Fast I/O services can be seen by contrasting the operation of Fast I/O system services to the $QIO system service as follows:

- The $QIO system service I/O status block (IOSB) is replaced by an I/O status area (IOSA) that is larger and quadword aligned. The transfer byte count returned in IOSA is 64 bits, and the field is aligned on a quadword boundary. Unlike the IOSB, which is optional, the IOSA is required.
- User data buffers must be aligned to a 512-byte boundary.
- All user process structures passed to the Fast I/O system services must reside in buffer objects. This includes the user data buffer and the IOSA.
- Only transfers that are multiples of 512 bytes are supported.
- Only the following function codes are supported: IO$_READVBLK, IO$_READLBLK, IO$_WRITEVBLK, and IO$_WRITELBLK.
- Only I/O to disk and tape devices is optimized for performance.
- No event flags are used with Fast I/O services. If application code must use an event flag in relation to a specific I/O, then the Event No Flag EFN (EFN$C_ENF) can be used. This event flag is a no-overhead EFN that can be used in situations when an EFN is required by a system service interface but has no meaning to an application.

  For example, Fast I/O services do not use EFNs, so the application cannot specify a valid EFN associated with the I/O to the $SYNCH system service with which to synchronize I/O completion. To resolve this issue, the application can call the $SYNCH system service passing as arguments: EFN$C_ENF and the address of the appropriate IOSA. Specifying EFN$C_ENF signifies to $SYNCH that no EFN is involved in the synchronization of the I/O. Once the IOSA has been written with a status and byte count,

return from the $SYNCH call occurs. The IOSA is now the central point of synchronization for a given Fast I/O (and is the only way to determine whether the asynchronous I/O is complete). For more information about EFN$C_ENF, see Section 4.19.

- To minimize argument passing overhead to these services, the $QIO parameters P3 through P6 are replaced by a single argument that is passed directly by the Fast I/O system services to device drivers. For disk-like devices, this argument is the media address (VBN or LBN) of the transfer. For drivers with complex parameters, this argument is the address of a descriptor or of a buffer specific to the device and function.
- Segmented transfers are supported by Fast I/O but are not fully optimized. There are two major causes of segmented transfers. The first is disk fragmenting. While this can be an issue, it is assumed that sites seeking maximum performance have eliminated the overhead of segmenting I/O due to fragmentation.

  A second cause of segmenting is issuing an I/O that exceeds the port's maximum limit for a single transfer. Transfers beyond the port maximum limit are segmented into several smaller transfers. Some ports limit transfers to 64K bytes. If the application limits its transfers to less than 64K bytes, then this type of segmentation should not be a concern.

5.1.4 Using Fast I/O Services

The three Fast I/O system services are:

- $IO_SETUP—Sets up an I/O.
- $IO_PERFORM[W]—Performs an I/O request.
- $IO_CLEANUP–Cleans up an I/O request.

5.1.4.1 Using Fandles

A key concept behind the operation of the new Fast I/O services is the file handle or **fandle**. A fandle is an opaque token that represents a ‘‘setup’’ I/O. A fandle is needed for each I/O outstanding from a process.

All possible setup, probing, and validation of arguments is performed off the mainline code path during application startup with calls to the $IO_SETUP system service. The I/O function, the AST address, the buffer object for the data buffer, and the IOSA buffer object are specified on input to $IO_SETUP service, and a fandle representing this setup is returned to the application.

To perform an I/O, the $IO_PERFORM system service is called, specifying the fandle, the channel, the data buffer address, the IOSA address, the length of the transfer, and the media address (VBN or LBN) of the transfer.

If the asynchronous version of this system service, $IO_PERFORM, is used to issue the I/O, then the application can wait for I/O completion using a $SYNCH specifying EFN$C_ENF and the appropriate IOSA. The synchronous form of the system service, $IO_PERFORMW, is used to issue an I/O and wait for it to complete. Optimum performance comes when the application uses AST completion; that is, the application does not issue an explicit wait for I/O completion.

To clean up a fandle, the fandle can be passed to the $IO_CLEANUP system service.

5.1.4.2 Modifying Existing Applications

Modifying an application to use the new Fast I/O services requires a few source-code changes. For example:

1. A programmer adds code to create buffer objects for the IOSAs and data buffers.
2. The programmer changes the application to use the new Fast I/O services. Not all $QIOs need to be converted. Only high-volume read/write I/O requests should be changed.

   A simple example is a ‘‘database writer’’ program, which writes modified pages back to the database. Suppose the writer can handle up to 16 simultaneous writes. At application startup, the programmer would add code to create 16 fandles by 16 $IO_SETUP system service calls.
3. In the main processing loop within the database writer, the programmer replaces the $QIO calls with $IO_PERFORM calls. Each $IO_PERFORM call uses one of the 16 available fandles. While the I/O is in progress, the selected fandle is unavailable for use with other I/O requests. The database writer is probably using AST completion and recycling fandle, data buffer, and IOSA once the completion AST arrives.

   If the database writer routine cannot return until all dirty buffers are written (that is, it must wait for all I/O completions), then $IO_PERFORMW can be used. Alternatively $IO_PERFORM calls can be followed by $SYNCH system service calls passing the EFN$C_ENF argument to await I/O completions.

   The database writer will run faster and scale better because I/O requests now use less CPU time.
4. When the application exits, a $IO_CLEANUP system service call is done for each fandle returned by a prior $IO_SETUP system service call. Then the buffer objects are deleted. Image rundown performs fandle and buffer object cleanup on behalf of the application, but it is good form for the application to clean up properly.

5.1.4.3 I/O Status Area (IOSA)

The central point of synchronization for a given Fast I/O is its IOSA. The IOSA replaces the $QIO system service’s IOSB argument and is larger. The byte count field in the IOSA is 64 bits and quadword aligned. Unlike the $QIO system service, Fast I/O services require the caller to supply an IOSA and require the IOSA to be part of a buffer object.

The IOSA context field can be used in place of the $QIO system service ASTPRM argument. The $QIO ASTPRM argument is typically used to pass a pointer back to the application on the completion AST to locate the user context needed for resuming a stalled user-thread. However, for the $IO_PERFORM system service, the ASTPRM on the completion AST is always the IOSA. Since there is no user-settable ASTPRM, an application can store a pointer to the user thread context for this I/O in the IOSA context field and retrieve the pointer from the IOSA in the completion AST.

5.1.4.4 $IO_SETUP

The $IO_SETUP system service performs the setup of an I/O and returns a unique identifier for this setup I/O, called a fandle, to be used on future I/Os. The $IO_SETUP arguments used to create a given fandle remain fixed throughout the life of the fandle. This has implications for the number of fandles needed in an application. For example, a single fandle can be used only for reads or only for writes. If an application module has up to 16 simultaneous reads or writes pending, then potentially 32 fandles are needed to avoid any $IO_SETUP calls during mainline processing.

The $IO_SETUP system service supports an expedite flag, which is available to boost the priority of an I/O among the other I/O requests that have been handed off to the controller. Unrestrained use of this argument is useless, because if all I/O is expedited, nothing is expedited. Note that this flag requires the use of ALTPRI and PHY_IO privilege.

5.1.4.5 $IO_PERFORM[W]

The $IO_PERFORM[W] system service accepts a fandle and five other variable I/O parameters for the high-performance I/O operation. The fandle remains in use to the application until the $IO_PERFORMW returns or if $IO_PERFORM is used until a completion AST arrives.

The CHAN argument to the fandle contains the data channel returned to the application by a previous file operation. This argument allows the application the flexibility of using the same fandle for different open files on successive I/Os. However, if the fandle is used repeatedly for the same file or channel, then an internal optimization with $IO_PERFORM is taken.

Note that $IO_PERFORM was designed to have no more than six arguments to take advantage of the *OpenMS Calling Standard*, which specifies that calls with up to six arguments can be passed entirely in registers.

5.1.4.6 $IO_CLEANUP

A fandle can be cleaned up by passing the fandle to the $IO_CLEANUP system service.

5.1.4.7 Fast I/O FDT Routine (ACP_STD$FASTIO_BLOCK)

Since $IO_PERFORM supports only four function codes, this system service does not use the generalized FDT dispatching that is contained in the $QIO system service. Instead, $IO_PERFORM uses a single vector in the driver dispatch table called DDT$PS_FAST_FDT for all the four supported functions. The DDT$PS_ FAST_FDT field is a FDT routine vector that indicates whether the device driver called by $IO_PERFORM is set up to handle Fast I/O operations. A nonzero value for this field indicates that the device driver supports Fast I/O operations and that the I/O can be fully optimized.

If the DDT$PS_FAST_FDT field is zero, then the driver is not set up to handle Fast I/O operations. The $IO_PERFORM system service tolerates such device drivers, but the I/O is only slightly optimized in this circumstance.

The OpenVMS disk and tape drivers that ship as part of OpenVMS Version 7.0 have added the following line to their driver dispatch table (DDTAB) macro:

```
FAST_FDT=ACP_STD$FASTIO_BLOCK,- ; Fast-IO FDT routine
```

This line initializes the DDT$PS_FAST_FDT field to the address of the standard Fast I/O FDT routine, ACP_STD$FASTIO_BLOCK.

If you have a disk or tape device driver that can handle Fast I/O operations, then you can add this DDTAB macro line to your driver. If you cannot use the standard Fast I/O FDT routine, ACP_STD$FASTIO_BLOCK, then you can develop your own based on the model presented in this routine.

5.1.5 Additional Information

For complete information about the following Fast I/O system services, see the *OpenVMS System Services Reference Manual: A–GETMSG* and *OpenVMS System Services Reference Manual: GETQUI–Z.*

$CREATE_BUFOBJ
$DELETE_BUFOBJ
$CREATE_BUFOBJ_64
$IO_SETUP
$IO_PERFORM
$IO_CLEANUP

To see an example program that demonstrates the use of buffer objects and the new Fast I/O system services, refer to the IO_PERFORM.C program in the SYS$EXAMPLES directory.

5.2 Fast Path

Fast Path is an optional, high-performance feature designed to improve I/O performance. By restructuring and optimizing class and port device driver code around high-volume I/O code paths, Fast Path creates a streamlined path to the device. Fast Path is of interest to any application where enhanced I/O performance is desirable. Two examples are database systems and real-time applications, where the speed of transferring data to disk is often a vital concern.

Using Fast Path features does not require source-code changes. Minor interface changes are available for expert programmers who want to maximize Fast Path benefits.

In OpenVMS Alpha Version 7.0, Fast Path only supports disk I/O for the CIXCD port. This port provides access to CI storage for XMI based systems.

Fast Path is not currently available on the OpenVMS VAX operating system.

5.2.1 Fast Path Features and Benefits

Fast Path achieves dramatic performance gains by reducing CPU time for I/O requests on both uniprocessor and SMP systems. These savings are on the order of 25% less CPU cost per I/O request on a uniprocessor and 35% less on a multiprocessor system. The performance benefits are produced by:

- Reducing code paths through streamlining for the case of high-volume I/O
- Substituting port-specific spinlocks for global I/O subsystem spinlocks
- Affinitizing an I/O request for a given port to a specific CPU

The performance improvement can best be seen by contrasting the current OpenVMS I/O scheme to the new Fast Path scheme. While transparent to an OpenVMS user, each disk and tape device is tied to a specific port interconnect. All I/O for a device is sent out over its assigned port. Under the current OpenVMS I/O scheme, a multiprocessor I/O can be initiated on any CPU, but I/O completion must occur on the primary CPU. Under Fast Path, all I/O for a given port is affinitized to a specific CPU, eliminating the requirement for completing the I/O on the primary CPU. This means that the entire I/O can

be initiated and completed on a single CPU. Since I/O operations are no longer split among different CPUs, performance increases as memory cache thrashing between CPUs decreases.

Fast Path also removes a possible SMP bottleneck on the primary CPU. If the primary CPU must be involved in all I/O, then once this CPU becomes saturated, no further increase in I/O throughput is possible. Spreading the I/O load evenly among CPUs in a multiprocessor system provides greater maximum I/O throughput on a multiprocessor system.

With most of the I/O code path executing under port-specific spinlocks and with each port assigned to a specific CPU, a scalable SMP model of parallel operation exists. Given multiple port and CPUs, I/O can be issued in parallel to a large degree.

5.2.2 Using Fast Path

This section describes how to use the FAST_PATH SYSGEN parameter to use Fast Path.

FAST_PATH

FAST_PATH is a SYSGEN parameter that enables (1) or disables (0) Fast Path performance features for all Fast Path capable ports. Fast Path is disabled by default.

Preferred CPU

Each Fast Path capable port is affinitized to a specific CPU called the **preferred CPU**. All I/O for all devices serviced by this port initiates and completes on the preferred CPU.

Processes issuing I/O to a port on the port's preferred CPU have an inherent advantage in that the overhead to affinitize the I/O to the preferred CPU can be avoided. An application process can use the $PROCESS_AFFINITY system service to affinitize itself to the preferred CPU of the device to which the majority of its I/O is sent. With proper attention to affinity, a process's execution need never leave the preferred CPU. This presents a scalable process and I/O scheme for maximizing multiprocessor system operation. Like most RISC systems, Alpha system performance is highly dependent on the performance of CPU memory caches. Process affinity and preferred CPU affinity are two keys to minimizing the memory stalls in the application and in the operating system, thereby maximizing multiprocessor system throughput.

IO_PREFER_CPUS

IO_PREFER_CPUS is a CPU bit mask that controls the initial assignment of Fast Path capable ports to CPUs. Assigning a Fast Path port to a CPU means that the CPU cannot be stopped with the STOP/CPU command. If you want to preserve the ability to stop certain CPUs even when Fast Path is enabled, use IO_PREFER_CPUS. IO_PREFER_CPUS specifies the CPUs that can serve as preferred CPUs and that can be assigned a Fast Path port by the default assignment algorithm. CPUs whose bit is clear in the IO_PREFER_CPUS bit mask are not assigned a Fast Path port and can be stopped. IO_PREFER_CPUS defaults to -1, which specifies that all CPUs are able to be assigned Fast Path ports.

The initial assignment spreads Fast Path ports evenly among available CPUs in a round-robin fashion, making sure that the primary CPU is the last CPU to receive a port. The primary CPU is slightly offloaded because it might be busy processing non-Fast Path I/O.

$QIO IO$_SETPRFPATH ! IO$M_PREFERRED_CPU

You can change the assignment of a Fast Path port to a CPU by issuing a $QIO IO$_SETPRFPATH (Set Preferred Path) to the port device, for example, PNA0. The IO$M_PREFERRED_CPU modifier must be set, and the $QIO argument P1 must be set to a 32-bit CPU bit mask with a bit set indicating the new preferred CPU. On return from the I/O, the port and its associated devices are all affinitized to a new preferred CPU. Note that explicitly setting the preferred CPU overrides any default assignment of Fast Path ports to CPUs. This interface allows you the flexibility to load balance I/O activity over multiple CPUs in an SMP system. This is important because I/O activity can change over the course of a day or week.

$GETDVI DVI$_PREFERRED_CPU & SDA SHOW DEVICE

For an application seeking optimal Fast Path benefits, you can code each application process to run on the preferred CPU where the majority of the process's I/O activity occurs. To identify the preferred CPU for any Fast Path-capable device when Fast Path is enabled, use the SDA command SHOW DEVICE to display the preferred CPU ID for a port or disk device.

Alternatively, the $GETDVI system service or the DCL F$GETDVI lexical function will return the preferred CPU for a given device or file. The $GETDVI system service item code is DVI$_PREFERRED_CPU and the F$GETDVI item code string argument is PREFERRED_CPU. The return argument is a 32-bit CPU bit mask with a bit set indicating the preferred CPU. A return argument containing a bit mask of zero indicates that no preferred CPU exists, either because Fast Path is disabled or the device is not a Fast Path capable device. The return argument is designed to serve as a CPU bit mask input argument to the $PROCESS_AFFINITY system service that can be used to affinitize an application process to the optimal preferred CPU.

A high-availability feature of VMSclusters is that dual-pathed devices automatically fail over to a secondary path, if the primary path becomes inoperable. Because a Fast Path device could fail over to another path or port, and thereby, to another preferred CPU, an application can occasionally reissue the $GETDVI in a timer thread to check that process affinity is optimal.

5.2.3 Fast Path Restrictions

Fast Path restrictions include the following:

- Only high-volume I/Os are optimized.

  Fast Path streamlines the operation of high-volume I/O. I/O that does not meet the definition of high-volume is not optimized. A high-volume Fast Path I/O is characterized as follows:

  1. A virtual, logical, or physical read or write I/O without special I/O modifiers.
  2. An I/O request that is less than 64K bytes in size.
  3. An I/O issued when all I/O resources exist that needed to perform the I/O.

- Send credits resource must be managed.

  Applications seeking maximum performance must ensure the availability of sufficient I/O resources.

The only I/O resource that a Fast Path user needs to be concerned about is send credits. Send credits are extended by DSA controllers to host systems and represent the maximum number of I/Os that can be outstanding at any given point in time. If an application sends an unlimited number of simultaneous I/Os to a controller, it is likely that some I/O will back up waiting for send credits. You can tell whether the send-credit limit is being exceeded by using the DCL command SHOW CLUSTER/CONTINUOUS, followed by an ADD CONNECTIONS, CR_WAIT command. Rapidly increasing credit-wait counts for the disk-class driver connections (a LOC_PROC_NAME name of VMS$DISK_CL_DRVR) is a sign that an application may be incurring send-credit waits.

To ensure sufficient send credits, some controllers, like the HSC and HSJ, allow the number of send credits to vary. However, not all controllers have this flexibility, and different controllers have different send-credit limits. The best workaround is to know your application access patterns and look for send credit waits. If the number of send credits is being exhausted on one node, then add another controller to spread the load over multiple controllers. An alternative is to rework the application to load balance controller activity throughout the cluster, spreading a given controller's disk load over multiple nodes and allowing an application to exceed the send credits allotted to one node.♦

6

New DECamds Features

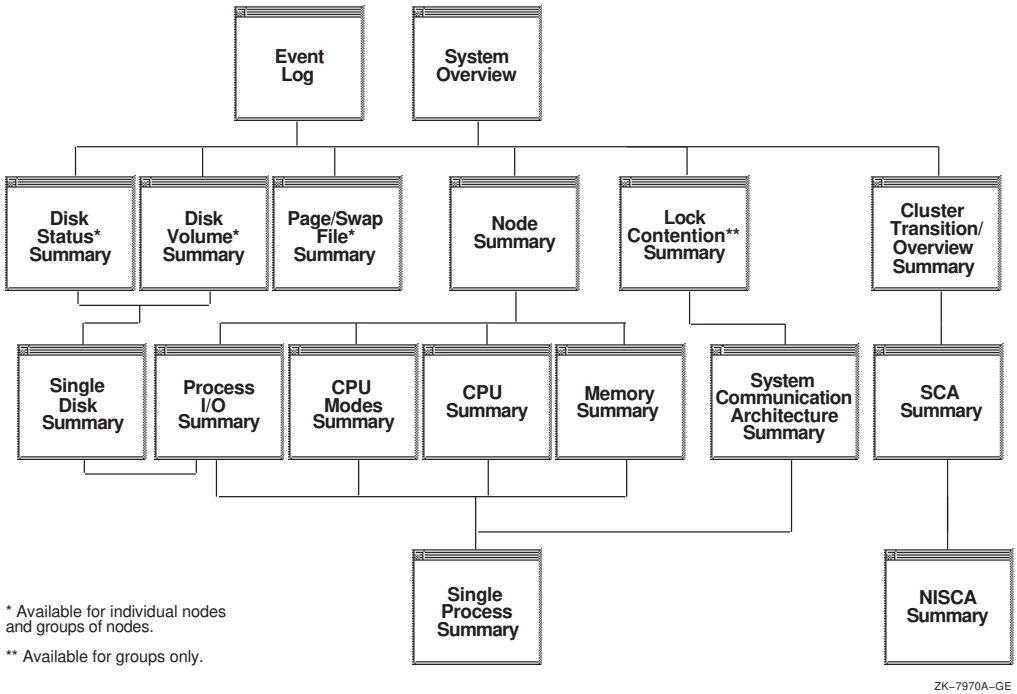
This chapter contains sections that explain the following new features of DECamds:

- New fields in the System Overview window: Number of Processes in CPU Queues, Operating System Version, Hardware Model
- New Single Disk Summary window: provides summary data about each node in the group in which a disk is available
- New cluster windows:
  - Cluster Transition/Overview Summary window: provides summary information about each node's membership in a VMScluster
  - System Communication Summary (SCA Summary) window: provides System Communication Architecture (SCA) information about a selected node's connection or connections to other nodes in a cluster
  - NISCA Summary window: provides summary information about the Network Interconnect System Communication Architecture (NISCA) protocol, which is responsible for carrying messages to other nodes in the cluster

These new features are documented in the Bookreader version of the Version 7.0 of the *DECamds User's Guide*.

Figure 6–1 shows where the new windows fit into the hierarchy of DECamds data windows.

Figure 6–1 DECamds Data Window Hierarchy

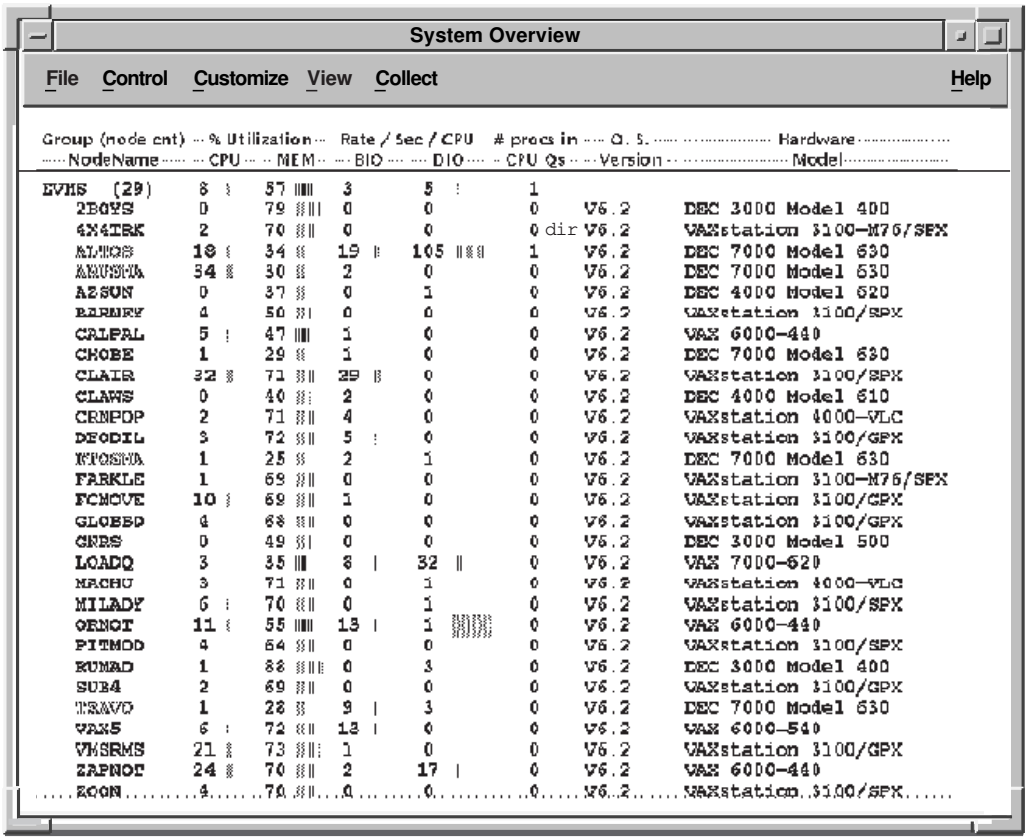


6.1 New Fields in the System Overview Window

Three new fields have been added to the System Overview window, which is described in Section 2.2 of the Bookreader version of the *DECamds User's Guide*.

The screen in Figure 6–2 includes the new fields in the System Overview window. This window shows the nodes that DECamds can currently reach and monitor.

Figure 6–2 System Overview Window



ZK–8543A–GE

Table 6–1 describes the new fields in the System Overview window.

Table 6–1 New Fields in the System Overview Window

Field	Description
# procs in CPU Qs (number of processes in CPU queues)	Represents the number of processes the Node Summary data collection found in the COM, COMO, MWAIT, and PWAIT CPU queues.
O.S. Version (version of the operating system)	Lists the currently loaded version of OpenVMS on the node being monitored (not the node doing the monitoring).
Hardware Model	Lists the hardware model of the node.

6.2 Single Disk Summary Window

The description of this new window follows Sections 3.3 and 3.4 in the Bookreader version of the *DECamds User's Guide*. These sections describe, respectively, the Disk Status Summary window and the Disk Volume Summary window. You can access the Single Disk Summary window from either window, as shown in Figure 6–1.

New DECamds Features
6.2 Single Disk Summary Window

The Single Disk Summary window shown in Figure 6–3 displays summary data about each node in the group in which a disk is available. This window is a node-by-node display of the data that is summarized in the Group Disk Status and Volume Summary windows. The values displayed are those you would see if you displayed Disk Status Summary or Disk Volume Summary for each node within the group.

You can use this display to determine both of the following:

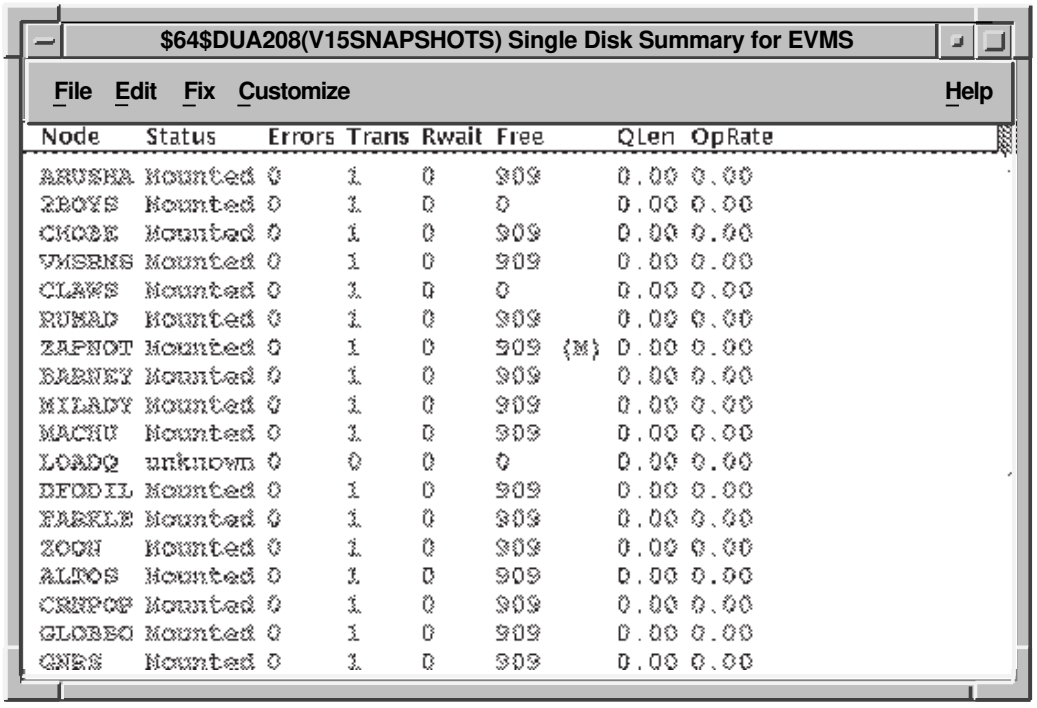
- Which node in the group has a disk with high I/O rates

  Determining which node has a high I/O rate to the disk is useful because you can sort by Direct I/O rate and learn which process or processes are causing the high I/O rates to the disk.

- If a disk is in a state inconsistent with other nodes

  Determining which node or nodes might be in an abnormal state is useful because you can then discover if, for some reason, one node believes that the disk is in the *MntVerify* or *CluTran* state, thus holding up processing in the cluster in which the node resides.

Figure 6–3 Single Disk Summary Window



ZK–8544A–GE

To open a Single Disk Summary window, follow these steps:

1. In the System Overview window, click MB3 on a group or node name.

   The system displays a pop-up menu.

2. Choose Display from the menu and Disk Status Summary (or Disk Volume Summary) from the submenu.

   The system displays the Disk Status Summary window (or Disk Volume Summary window).

3. In the Disk Status Summary window (or Disk Volume Summary window), click MB3 on a device name.

   The system displays a pop-up menu.

4. Choose Display Disk.

   The system displays the Single Disk Summary window.

Table 6–2 lists the Single Disk Summary Window data fields.

Table 6–2 Data Items in the Single Disk Summary Window

Data Item	Description
Node	Name of the node
Status	Status of the disk: mounted, online, offline, and so on
Errors	Number of errors on the disk
Trans	Number of currently-in-progress file system operations on the disk (number of open files on the volume)
Rwait	Indication of an I/O stalled on the disk
Free	Count of free disk blocks on the volume An (M) after the free block count indicates this node holds the lock on the volume that DECamds uses to obtain the true free block count on the volume. Other nodes might not have accessed the disk, so their free block count might not be up to date.
QLen	Average number of operations in the I/O queue for the volume
OpRate	Count of rate of change to operations on the volume

Note

When you click on an item, DECamds temporarily stops updating the window for 15 seconds or until you choose an item from a menu.

From the Single Disk Summary window, you can display the Process I/O Summary window. To do so, follow these steps:

1. Click MB3 anywhere on a node line.

   The system displays a pop-up menu.

2. Choose Display Process I/O Summary.

   The system displays the Process I/O Summary window.

See the *DECamds User's Guide* for information about the Process I/O Summary window.

6.3 New Cluster Windows

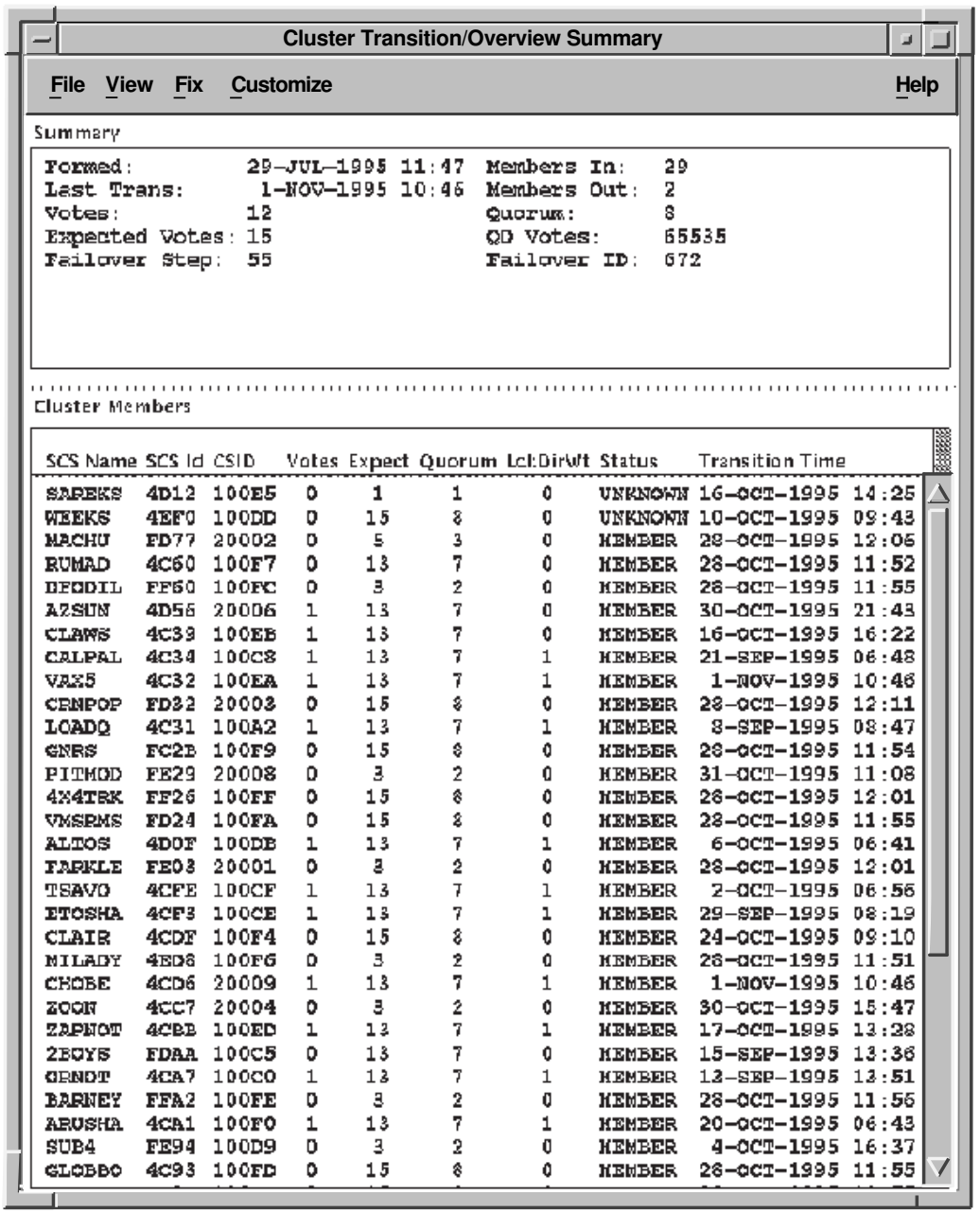
The descriptions of the three new cluster windows shown in the diagram in Figure 6–1 follow the last window described in Chapter 3 of the Bookreader version of the *DECamds User's Guide*.

For conceptual information about cluster data displayed in the windows, refer to *VMScluster Systems for OpenVMS*.

6.3.1 Cluster Transition/Overview Summary Window

The Cluster Transition/Overview Summary window shown in Figure 6–4 displays information about each node in a VMScluster. This window is very similar to System Overview window; however, this window lists only one cluster for each set of nodes in a cluster, while the System Overview window lists all the nodes and the user-defined groups these nodes are in.

Figure 6–4 Cluster Transition/Overview Summary Window



ZK–8545A–GE

The window displays summary information as well as information about individual nodes: System Communication Services (SCS) name, SCS ID, Cluster System ID, Votes, Lock Directory Weight value, cluster status, and last transition time.

The data items shown in the window correspond to data that the Show Cluster utility displays for the SYSTEM and MEMBERS classes. A status field display of "unknown" usually indicates that DECamds is not communicating with the node.

To open a Cluster Transition/Overview Summary window, follow these steps:

1. In the System Overview window, click MB3 on a node line.

   The system displays a pop-up menu.

2. Choose Display from the menu and Cluster Transition Summary from the submenu.

   The system displays the Cluster Transition/Overview Summary window.

Note

The Cluster Transition Summary menu option is not available for nodes that are not in the cluster, nor is it available from group lines in the display.

6.3.1.1 Data Displayed

The Cluster Transition/Overview window has two panel displays:

- Summary (top) panel: displays VMScluster summary information.
- Cluster Members (bottom) panel: lists each node in the cluster.

Table 6–3 describes the Summary panel data fields.

Table 6–3 Data Items in the Summary Panel of the Cluster Transition/Overview Summary Window

Data Item	Description
Formed	Date and time the VMScluster was formed.
Last Trans	Date and time of the most recent VMScluster state transition.
Votes	Total number of quorum votes being contributed by all cluster members and quorum disk.
Expected Votes	Number of votes expected to be contributed by all members of the cluster as determined by the connection manager. This value is based on the maximum of EXPECTED_VOTES and the maximized value of VOTES.
Failover Step	Current failover step index.
Members In	Number of members of the cluster DECamds has a connection to.
Members Out	Number of members of the cluster DECamds either has no connection to or has lost connection to.
Quorum	Number of votes required to keep cluster above quorum.

(continued on next page)

Table 6–3 (Cont.) Data Items in the Summary Panel of the Cluster Transition /Overview Summary Window

Data Item	Description
QD Votes	Number of votes given to Quorum Disk. A value of 65535 means there is no Quorum Disk.
Failover ID	Failover Instance Identification.

Table 6–4 describes the Cluster Members panel data fields.

Table 6–4 Data Items in the Cluster Members Panel of the Cluster Transition /Overview Summary Window

Data Item	Description
SCS Name	System Communication Services name for the node (system parameter SCSNODE)
SCS Id	System Communication Services identification for the node (system parameter SCSYSTEMID)
CSID	Cluster System Identification
Votes	Number of votes the member contributes
Expect	Expected votes to be contributed as set by the EXPECTED_VOTES
Quorum	Recommended quorum value derived from the expected votes
LckDirWt	Lock Manager distributed directory weight as determined by the LCKDIRWT system parameter
Status	Current cluster member status: MEMBER, UNKNOWN, or BRK_NON (break_non-member)
Transition Time	Time cluster member had last transition

6.3.1.2 Notes About Data Display

The following are notes about the display of data in the window:

- No highlighting conventions are used in the window; all data items are displayed in bright mode.
- You cannot filter out any data.
- The data items in the window are sorted on an "as-found" basis. You cannot change the sort criteria.
- When you click on an item, DECamds temporarily stops updating the window for 15 seconds or until you choose an item from a menu.
- You can change collection intervals.

6.3.1.3 New Event in Window

The following new event has been created for the display in this window:

```
LOVOTE, 'node' VOTES count is close to or below QUORUM
```

DECamds signals this event when the difference between the cluster's QUORUM and VOTES values *is less than* the threshold for the event. The default threshold for the event is 1.

6.3.1.4 From This Window...

From this window, you can do the following:

- Double-click MB1 on a line to open a Node Summary display.
- Highlight a node and select a menu option to display either of the following:
  - Node Summary display of nodes that DECamds recognizes. DECamds ignores nodes that are unknown or break_non-member.
  - SCA Summary display of nodes that DECamds recognizes. DECamds ignores nodes that are unknown or break_non-member.
- Perform the Cluster Quorum Adjustment fix.

  This fix forces a cluster quorum adjustment on the entire OpenVMS cluster on which the fix is run.

  To perform the fix, first select the Fix option on the menu bar. Then the Quorum option on the menu displayed. DECamds moves through the cluster membership to find the first member node it can communicate with and performs a Quorum Adjustment fix on that node.

6.3.2 SCA Summary Window

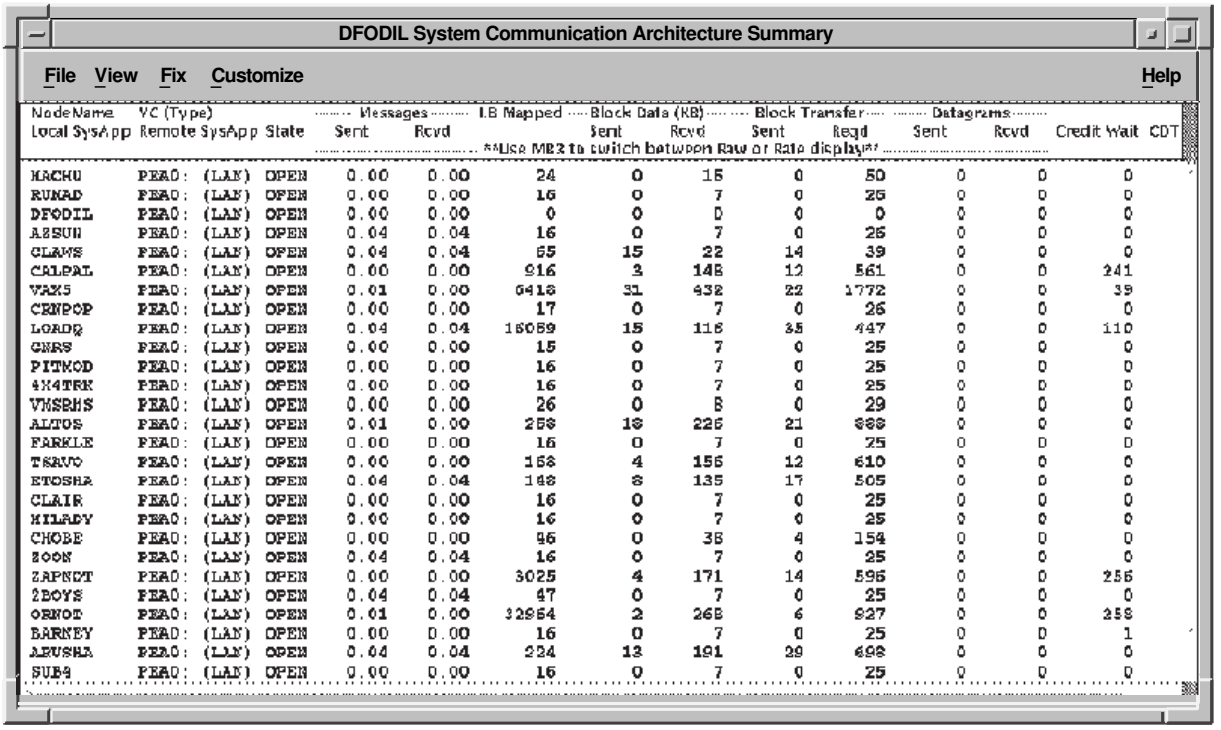
The System Communication Architecture Summary (SCA Summary) window shown in Figure 6–5 displays information about a selected node's virtual circuits and connections to other nodes in a cluster. (The display represents the view one node has of other nodes in the cluster.) More than one type of virtual circuit indicates that more than one path to the remote node exists.

Each line in the window shows either a summary of all system applications (SysApps) using the virtual circuit communication or the communication on the connection between a local and a remote SysApp. The data displayed in the window is similar to the information that the Show Cluster utility displays for the CIRCUITS, CONNECTIONS, and COUNTERS classes. Unlike Show Cluster, however, this display shows only SCA connections to other OpenVMS nodes; it does not show SCA connections to the Disk Storage Architecture (DSA) or to devices such as FDDI or DSSI disk controllers.

By clicking MB3 on a node name and choosing View SysApps from the pop-up menu, you can display the system applications that are using virtual circuits. This option expands the list below a virtual circuit to all the system applications that contribute to that virtual circuit. (The SysApp lines are dimmed and right-justified.)

To hide the display of system applications, you can click MB3 and choose Hide SysApps from the pop-up menu.

Figure 6–5 SCA Summary Window



ZK–8546A–GE

You can click MB3 on the data to the right of "State" to display a menu allowing you to toggle between Raw and Rate data.

To open an SCA Summary window, follow these steps:

1. In the Cluster Transition/Overview Summary window, click MB3 on an SCS name.

   The system displays a pop-up menu.

2. Choose Display SCA Summary.

   The system displays the System Communication Architecture (SCA) Summary window.

Table 6–5 describes the SCA Summary window data fields.

Table 6–5 Data Items in the SCA Summary Window

Data Item	Description
NodeName	SCS name of the remotely connected node.
VC(Type)	The virtual circuit being used and its type.
State	The state of the virtual circuit connection.
Messages	Relatively small data packets sent and received between nodes for control information.
Block Transfer	Fields listing the count of the number of block data transfers and requests initiated.

(continued on next page)

Table 6–5 (Cont.) Data Items in the SCA Summary Window

Data Item	Description
KB Mapped	Field listing the number of kilobytes mapped for block data transfer. Note: This field is available in RAW format only.
Block Data (KB)	Fields listing in kilobytes the data transferred via block data transfer.
Datagrams	Number of unacknowledged messages sent between virtual circuits.
Credit Wait	Number of times the connection had to wait for a send credit.
BDT Wait	Number of times the connection had to wait for a buffer descriptor.
Local SysApp	Name of the local system application using the virtual circuit.
Remote SysApp	Name of the remote system application being communicated to.

6.3.2.1 Notes About Data Display

The following are notes about the display of data in the window:

- The window does not follow highlighting conventions: virtual circuit lines are displayed brightly and are left-aligned; SysApp lines are dimmed and are indented by a column.
- You cannot filter out any data.
- The data items in the window are sorted on an "as-found" basis. You cannot change sort criteria at this time.
- For messages, the default is the display of rate data; raw data is the default for all other types of data.
- You can change collection intervals.

6.3.2.2 New Event in Window

The following new event has been created for the display in this window:

```
LOSTVC, <node> lost virtual circuit (<string>) to node <node>
```

DECamds signals this event when a virtual circuit between two nodes has been lost. This loss might be due either to a cluster node crashing or to cluster problems that caused the virtual circuit to close.

6.3.2.3 From This Window...

From this window, you can display the Network Interconnect System Communication Architecture (NISCA) Summary window. DECamds displays one window per virtual circuit provided the virtual circuit is running over a PEA0: device. See Section 6.3.3 for instructions.

6.3.3 NISCA Summary Window

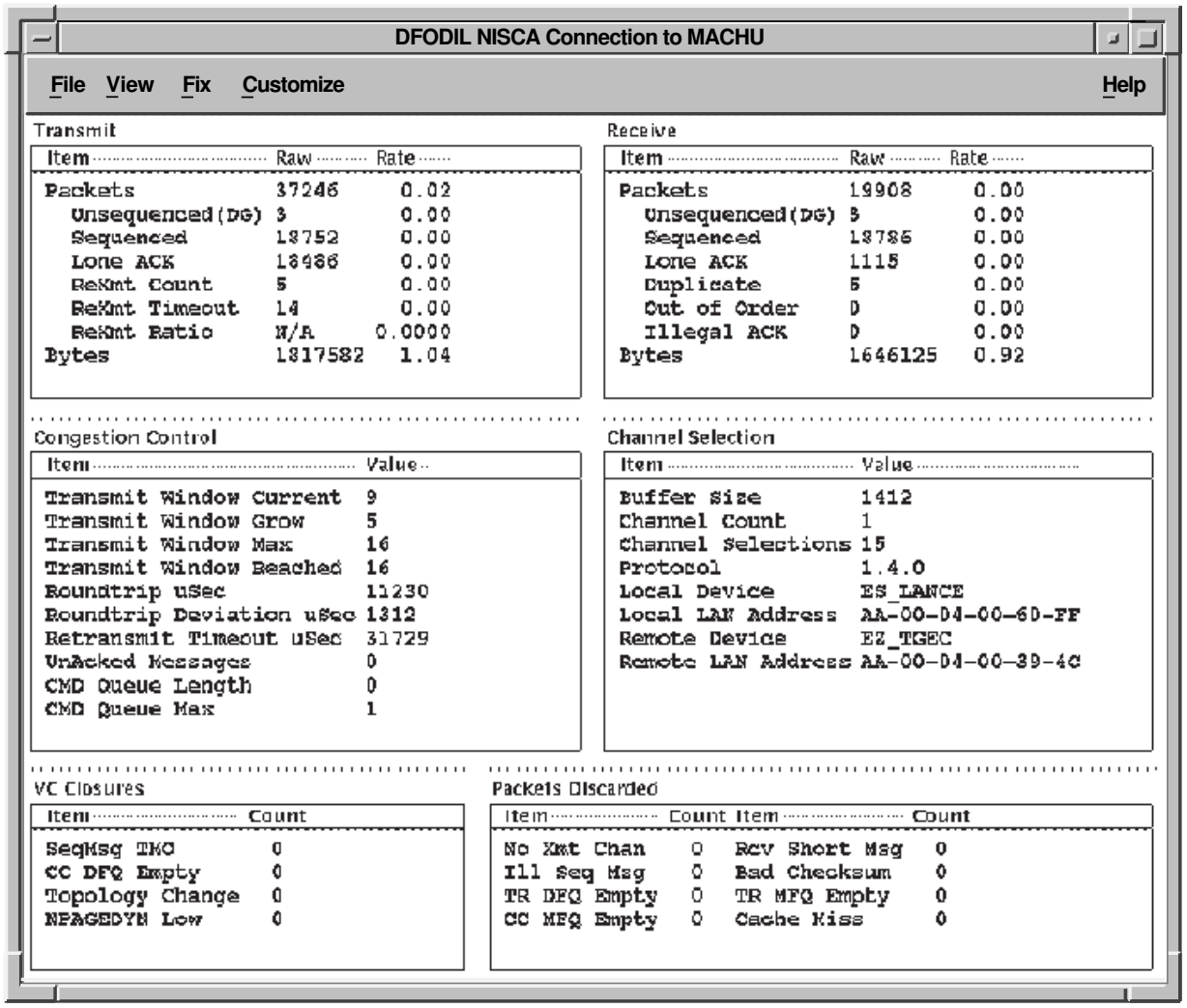
The Network Interconnect System Communication Architecture (NISCA) is the transport protocol responsible for carrying messages such as disk I/Os and lock messages across Ethernet and FDDI LANs to other nodes in the cluster. More detailed information about the protocol is in *VMScluster Systems for OpenVMS* in the OpenVMS documentation set.

The NISCA Summary window shown in Figure 6–6 displays detailed information about the LAN (Ethernet or FDDI) connection between two nodes. DECamds displays one window per virtual circuit provided the virtual circuit is running over a PEA0: device.

This window is designed to view statistics in real time and to troubleshoot problems found in the NISCA protocol. The window is intended primarily as an aid to diagnosing LAN-related problems. Section F.4 in Appendix F of the *VMScluster Systems for OpenVMS* describes the parameters shown in this window and tells how to use them to diagose LAN-related cluster problems.

The window provides the same information as the OpenVMS System Dump Analyzer (SDA) command SHOW PORTS/VC=VC_nodex. (Nodex is a node in the cluster; the system defines VC-nodex after a SHOW PORTS command is issued from SDA.)

Figure 6–6 NISCA Summary Window



ZK–8547A–GE

To open an NISCA Summary window, follow these steps:

1. In the SCA Summary window, click MB3 on a row with the PEA0: Virtual Circuit.

   The system displays a pop-up menu.

2. Choose View SysApps.

The system displays an expanded list below the node name.

3. Click MB3 on a SysApps node.

   The system displays a pop-up menu.

4. Choose Display NISCA.

   The system displays the NISCA Summary window.

Note

If the Display NISCA option is dimmed, the NISCA protocol is not running for that system application.

6.3.3.1 Data Displayed

Panels in the NISCA Summary window contain the data described in the following tables.

Table 6–6 lists data items displayed in the Transmit Panel, which contains data packet transmission information.

Table 6–6 Data Items in the Transmit Panel

Data Item	Description
Packets	Number of packets transmitted through the virtual circuit to the remote node, including both sequenced and unsequenced (channel control) messages, and lone acknowledgments.
Unsequenced (DG)	Count and rate of the number of unsequenced datagram packages transmitted.
Sequenced	Count and rate of the number of sequenced packages transmitted. Sequenced messages are used for application data.
Lone ACK	Count and rate of the number of lone acknowledgments.
ReXmt Count	Number of packets retransmitted. Retransmission occurs when the local node does not receive an acknowledgment for a transmitted packet within a predetermined timeout interval.
ReXmt Timeout	Number of retransmission timeouts that have occurred.
ReXmt Ratio	Ratio of ReXmt Count current and past to the current and past number of sequenced messages sent.
Bytes	Count and rate of the number of bytes transmitted through the virtual circuit.

Table 6–7 describes data items displayed in the Receive Panel, which contains data packet reception information.

Table 6–7 Data Items in the Receive Panel

Data Item	Description
Packets	Number of packets transmitted through the virtual circuit to the remote node, including both sequenced and unsequenced (channel control) messages, and lone acknowledgments.

(continued on next page)

Table 6–7 (Cont.) Data Items in the Receive Panel

Data Item	Description
Unsequenced (DG)	Count and rate of the number of unsequenced packages received.
Sequenced	Count and rate of the number of sequenced packages received. Sequenced messages are used for application data.
Lone ACK	Count and rate of the number of lone acknowledgments.
Duplicate	Number of redundant packets received by this system.
Out of Order	Number of packets received out of order by this system.
Illegal Ack	Number of illegal acknowledgments received.
Bytes	Count and rate of the number of bytes received through the virtual circuit.

Table 6–8 describes data items displayed in the Congestion Control Panel, which contains transmit congestion control information.

The values in the panel list the number of messages that can be sent to the remote node before receiving an acknowledgment and the retransmission timeout.

The system parameter PEDRIVER varies the pipe quota and the timeout value to control the amount of network congestion.

Table 6–8 Data Items in the Congestion Control Panel

Data Item	Description
Transmit Window Current	Current value of the pipe quota (transmit window). After a timeout, the pipe quota is reset to 1 to decrease congestion and is allowed to increase quickly as acknowledgments are received.
Transmit Window Grow	The slow growth threshold: size at which the rate of increase is slowed to avoid congestion on the network again.
Transmit Window Max	Maximum value of pipe quota currently allowed for the virtual circuit based on channel limitations.
Transmit Window Reached	Number of times the entire transmit window was full. If this number is small as compared with the number of sequenced messages transmitted, the local node is not sending large bursts of data to the remote node.
Roundtrip uSec	Average roundtrip time for a packet to be sent and acknowledged. The value is displayed in microseconds.
Roundtrip Deviation uSec	Average deviation of the roundtrip time. The value is displayed in microseconds
Retransmit Timeout uSec	Value used to determine packet retransmission timeout. If a packet has not received either an acknowledging or responding packet, it is assumed that the sent packet is lost and it will be resent.
UnAcked Messages	Number of unacknowledged messages.
CMD Queue Length	Current length of all command queues.
CMD Queue Max	Maximum number of commands in queues so far.

Table 6–9 describes data items displayed in the Channel Selection Panel, which contains channel selection information.

Table 6–9 Data Items in the Channel Selection Panel

Data Item	Description
Buffer Size	Maximum PPC data buffer size for this virtual circuit
Channel Count	Number of channels connected to this virtual circuit
Channel Selections	Number of channel selections performed
Protocol	NISCA Protocol version
Local Device	Name of the local device that the channel uses to send and receive packets
Local LAN Address	Address of the local LAN device that performs sends and receives
Remote Device	Name of the remote device that the channel uses to send and receive packets
Remote LAN Address	Address of the remote LAN device performing the sends and receives

Table 6–10 describes data items displayed in the VC Closures Panel, which contains information about the number of times a virtual circuit has closed for a particular reason.

Table 6–10 Data Items in the VC Closures Panel

Data Item	Description
SeqMsg TMO	Number of sequence transmit timeouts
CC DFQ Empty	Number of times the channel control DFQ was empty
Topology Change	Number of times PEDRIVER performed a failover from FDDI to Ethernet, necessitating the closing and reopening of the virtual circuit
NPAGEDYN Low	Number of times the virtual circuit was lost because of a pool allocation failure on the local node

Table 6–11 lists data items displayed in the Packets Discarded Panel, which contains information about the number of times packets were discarded for a particular reason.

Table 6–11 Data Items in the Packets Discarded Panel

Data Item	Description
No Xmt Chan	Number of times there was no transmit channel
Ill Seq Msg	Number of times an illegal sequenced message was received
TR DFQ Empty	Number of times the Transmit DFQ was empty
CC MFQ Empty	Number of times the Channel Control MFQ was empty
Rcv Short Msg	Number of times a short transport message was received
Bad Checksum	Number of times there was a checksum failure
TR MFQ Empty	Number of times the Transmit MFQ was empty
Cache Miss	Number of messages that could not be placed in the cache

6.3.3.2 Notes About Data Display

The following are notes about the display of data in the window:

- No highlighting conventions are used within the NISCA Summary window.
- You cannot sort or filter the data displayed in this window.
- You can change collection intervals.

A

New OpenVMS System Messages

This release includes new or changed messages for the following OpenVMS facilities:

- BUGCHECK, System Bugcheck
- LAT, LAT Facility
- LINK, Linker Utility
- NETWRK, SET/SHOW/START/STOP NETWORK Commands
- SDA, System Dump Analyzer
- SET, SET Command, and SET Utility
- SYSTEM, System Services

A.1 List of Messages

This section alphabetically lists and describes messages that have been added or changed for this release. You can access online descriptions of these and other OpenVMS system messages by using the online Help Message utility. For information about the HELP/MESSAGE command and qualifiers, see DCL help (type HELP HELP/MESSAGE at the DCL prompt) or refer to *OpenVMS System Messages: Companion Guide for Help Message Users*.

Bookreader Version Only

To access the system messages, click on the Topic button.

ABORT, fatal error encountered; operation terminated

Facility: NETWRK, SET/SHOW/START/STOP NETWORK Commands

Explanation: The fatal error reported in an accompanying message caused the command to fail.

User Action: Correct the problem and retry the operation.

ALRDYCOMP, data in this dump file is already compressed; performing regular copy

Facility: SDA, System Dump Analyzer

Explanation: You specified /COMPRESS, but data in this dump file is already compressed. SDA is performing a regular copy.

User Action: None.

New OpenVMS System Messages
A.1 List of Messages

ALRDYDCMP, data in this dump file is already decompressed; performing regular copy

Facility: SDA, System Dump Analyzer

Explanation: You specified /DECOMPRESS, but data in this dump file is already decompressed. SDA is performing a regular copy.

User Action: None.

ARG_GTR_32_BITS, argument is not 32-bit sign-extended value

Facility: SYSTEM, System Services

Explanation: An attempt was made to pass a 64-bit virtual address to a service that is equipped to handle only 32-bit virtual addresses.

User Action: Ensure that all service arguments are 32-bit, sign-extended, virtual addresses. No P2 space addresses are allowed.

BADFANDLE, invalid fandle

Facility: SYSTEM, System Services

Explanation: The application has supplied an illegal fandle (Fast-I/O handle) to the system.

User Action: If possible, correct the error in the application program.

BADIOSADR, IOSA is not within buffer object

Facility: SYSTEM, System Services

Explanation: The application has supplied an illegal I/O Status Area (IOSA) address.

User Action: If possible, correct the error in the application program.

BITMAPERR, virtual address range %X'starting-address' to %X'ending-address' writes outside demand zero bitmap address range %X'starting-address' to %X'ending-address'

Facility: LINK, Linker Utility

Explanation: This is an internal error. The linker has written to what it perceives is a portion of the image binary proper but it did not allocate the address range (displayed in hexadecimal radix in the message) within the demand zero bitmap.

User Action: Contact a Digital support representative.

CANTOPEN, cannot open file 'file-name'

Facility: NETWRK, SHOW NETWORK Command

Explanation: An error was encountered while attempting to open the specified file for output.

User Action: Ensure that the file exists in the current directory and that you correctly specify the file name.

CHANVIO, the specified channel is not assigned or was assigned from a more privileged access mode

Facility: SYSTEM, System Services

Explanation: The specified channel is not assigned or has already been assigned to a more privileged access mode.

User Action: Ensure that the correct channel is specified to the system service.

COMPRESSFAIL, unable to compress/decompress the data in this dump file (internal error)

Facility: SDA, System Dump Analyzer

Explanation: An internal consistency check failed while processing an SDA COPY/COMPRESS or COPY/DECOMPRESS command.

User Action: Contact a Digital support representative and provide a copy of the dump file you were copying.

DEFLATE, copying dump data in compressed format

Facility: SDA, System Dump Analyzer

Explanation: As requested, SDA is copying the dump data in compressed format.

User Action: None.

DUPLICATE, specified network already exists

Facility: NETWRK, SET NETWORK Command

Explanation: The specified product is already registered.

User Action: You need not register this product again. If you wish to modify the product information, use the SET NETWORK/UPDATE command.

EMULATED, an instruction not implemented on this processor was emulated at PC='location', PS='xxxxxxxx'

Facility: SYSTEM, System Services

Explanation: When an image that was compiled for use with a new processor was executed on an older processor, the processor did not recognize a new instruction. The unrecognized instruction (located at the PC address) has been emulated in the software. The emulator is slow, so the image is not running at its best speed. The instruction emulator issues this message the first five times this condition occurs.

User Action: Contact the program's supplier for the version that runs most efficiently on your processor. If the code includes instructions designed for new processors, it will execute fastest on a new processor. If you have an old processor, it is best to have code containing new instructions compiled specifically for the old processor. The slowest alternative is to run a new instruction that must be emulated on an old processor.

FANDLEBUSY, fandle is already in use

Facility: SYSTEM, System Services

Explanation: An application has attempted to use one fandle for two Fast-I/O operations.

User Action: If possible, correct the error in the application program.

FULL, cannot add; number of network services is at capacity

Facility: NETWRK, SET NETWORK Command

Explanation: The maximum of 128 network services is already registered.

User Action: Before you can register more network services, you must remove one or more network services using the SET NETWORK/REMOVE command.

GBLSEC_MISMATCH, global section type does not match service called

Facility: SYSTEM, System Services

Explanation: A global section type of this name already exists and it does not match the type of section to be mapped.

User Action: Ensure that the correct global section name is specified.

HWRPB_SUSPECT, HWRPB checksum incorrect; continuing anyway

Facility: SDA, System Dump Analyzer

Explanation: While analyzing a full memory dump, SDA detected an incorrect checksum in the hardware restart parameter block (HWRPB).

User Action: None. SDA will attempt to continue reading the dump file. If this is a common occurrence, contact a Digital support representative and supply a copy of the dump being analyzed.

ILLBUFOBJ, buffer object handle is not valid

Facility: SYSTEM, System Services

Explanation: An application has passed an illegal buffer object handle to a system service that requires a valid buffer object handle.

User Action: If possible, correct the error in the application program.

ILLMODIFIER, I/O function modifier is not permitted

Facility: SYSTEM, System Services

Explanation: An application has requested an unsupported function code modifier with an otherwise valid Fast-I/O function. Each device driver supports only a few, if any, I/O function code modifiers for Fast-I/O.

User Action: If possible, correct the error in the application program. Alternatively, move the data to be accessed to a device whose driver supports the function code modifier for Fast-I/O.

ILLRELPAG, invalid relative page argument specified

Facility: SYSTEM, System Services

Explanation: The specified relative page argument is either larger than the highest page number within the section or is not a valid 32-bit physical page frame number.

User Action: Ensure that the correct relative page argument is associated with the specified page section.

INCONSISTENT, logical names structure left in inconsistent state

Facility: NETWRK, SET NETWORK Command

Explanation: An error that occurred while registering new services has disrupted the logical names structure and caused data to become corrupted.

User Action: Deregister all network services using the DCL command SET NETWORK/REMOVE. Then reregister the network services by executing the SYS$STARTUP:SYS$NET_SERVICES.COM command.

INFLATE, copying dump data in decompressed format

Facility: SDA, System Dump Analyzer

Explanation: As requested, SDA is copying the dump data in decompressed format.

User Action: None.

INVNODEUID, invalid node UID

Facility: LAT, LAT Facility

Explanation: A LAT connection was attempted, but the remote node returned a unique node identifier that was different than expected.

User Action: Retry the connection. If this error persists, look for multiple nodes on the network having the same LAT node name and change one of the names to a different unique node name.

INVREGID, no region matches the supplied ID

Facility: SDA, System Dump Analyzer

Explanation: No region matches the ID you specified in a SHOW PROCESS /RDE or SHOW PROCESS/PAGE/RDE command.

User Action: Correct the ID and retry the command.

IVACMODE, invalid access mode specified

Facility: SYSTEM, System Services

Explanation: The specified access mode is not of high enough privilege to create address space in the specified region.

User Action: Ensure that the correct access mode is associated with the specified region.

IVPORT, invalid port name

Facility: LAT, LAT Facility

Explanation: An attempt was made in LATCP to change a port characteristic. However, the specified port is not a LAT device.

User Action: Retry the command using a valid LAT device name.

IVREGFLG, invalid region flag specified

Facility: SYSTEM, System Services

Explanation: Either an invalid combination of flags was specified or reserved bits in the flags argument were set.

User Action: Check the flags argument and ensure that only legal flags are set in a valid combination.

IVREGID, invalid region id specified

Facility: SYSTEM, System Services

Explanation: The specified region does not exist.

User Action: Ensure that a proper region identification is passed to the system service.

IVREGPROT, invalid region protection specified

Facility: SYSTEM, System Services

Explanation: An invalid protection was specified to the region creation system service.

User Action: Ensure that the correct protection argument is passed to the service.

IVVAFLG, invalid virtual address flag specified

Facility: SYSTEM, System Services

Explanation: Either an invalid combination of flags was specified or reserved bits in the flags argument were set.

User Action: Check the flags argument and ensure that only legal flags are set in a valid combination.

LEN_NOTBLKMULT, specified length is not a multiple of virtual disk blocks

Facility: SYSTEM, System Services

Explanation: The specified length is not an even multiple of virtual disk blocks.

User Action: Ensure that the length passed to the system service is an even multiple of disk blocks.

LEN_NOTPAGMULT, specified length is not a multiple of CPU-specific pages

Facility: SYSTEM, System Services

Explanation: The specified length is not an even multiple of CPU-specific pages.

User Action: Ensure that the length passed to the system service is an even multiple of CPU-specific pages.

NOACCESS, process not accessible (swapped out or suspended)

Facility: SDA, System Dump Analyzer

Explanation: The process specified in a SHOW PROCESS or SET PROCESS command is inaccessible either because it was swapped out of the balance set or suspended (when using the ANALYZE/SYSTEM command) or it was swapped out of the balance set when the system failed (when using the ANALYZE/CRASH command).

User Action: None.

NOAVAILABLE, network service information is unavailable at this time

Facility: NETWRK, SHOW NETWORK Command

Explanation: No network services are registered.

User Action: Have the system manager verify that the SYS$STARTUP:SYS$NET_SERVICES.COM command is called from the SYS$STARTUP:SYSTARTUP_VMS.COM command procedure.

NOBUFOBJID, requires rights identifier VMS$BUFFER_OBJECT_USER

Facility: SYSTEM, System Services

Explanation: The program attempted to create a buffer object from user mode without having the VMS$BUFFER_OBJECT_USER identifier required for this privileged operation.

User Action: Either write the application to create buffer objects from an inner mode, or enable the VMS$BUFFER_OBJECT_USER identifier for the process.

If you have been granted the VMS$BUFFER_OBJECT_USER identifier, you may need to use this DCL command to enable it:

```
$ SET RIGHTS_LIST /ENABLE VMS$BUFFER_OBJECT_USER
```

If you have not been granted the VMS$BUFFER_OBJECT_USER identifier, this command will fail and you must see your system manager. If the system manager chooses to grant you the identifier, he or she can do so by entering this AUTHORIZE command:

```
UAF> GRANT/IDENT/ATTR=DYNAMIC VMS$BUFFER_OBJECT_USER username
```

Once the system manager has granted you the identifier, you must log out and then log back in to pick up the identifier. For more information about Authorize utility commands, see the *OpenVMS System Manager's Manual*.

NOCBMAP, compression block map unreadable for compressed dump data

Facility: SDA, System Dump Analyzer

Explanation: The header of the dump file specified in the ANALYZE /CRASH command indicates that the dump file contains compressed data, but SDA was unable to locate a compression block map for decoding the data. Therefore, the dump is unreadable.

User Action: None.

NOCCBBUFFOBJ, unable to lock down CCB into buffer object

Facility: SYSTEM, System Services

Explanation: The Fast-I/O system services are unable to make a buffer object out of certain sections of the caller's P1 space.

User Action: The application has encountered a problem in the user environment. Execute the DCL command SHOW MEMORY/BUFFER_OBJECT and retain the output. Obtain the application source code, if possible, and contact a Digital support representative.

NOCONTROL, control characters are not permitted

Facility: NETWRK, SET NETWORK Command

Explanation: A control character was included in the text strings.

User Action: Remove all control characters from the text strings.

NOIMSEM, Process is not multithreaded; no semaphore present

Facility: SDA, System Dump Analyzer

Explanation: The specified process is not a multithreaded process; therefore, it does not have any process-specific inner mode semaphore data.

User Action: None.

NOMOREREG, no more regions found

Facility: SYSTEM, System Services

Explanation: No region was found at an address higher than that specified in the starting virtual address argument specified in conjunction with the VA$_NEXT_REGSUM_BY_VA wildcard function code.

User Action: None. This return code signals the end of the wildcard region search.

NORATINGIMAGE, LAT rating image not loaded

Facility: LAT, LAT Facility

Explanation: An attempt was made to start the LAT ancillary control process (LATACP), but the LAT$RATING image was not loaded.

User Action: Be sure to start LAT by executing the LAT$STARTUP command procedure to access the support startup files supplied by Digital. The LAT$STARTUP command procedure executes other command procedures, including LAT$CONFIG, which loads the LAT$RATING image.

NORDACC, read access denied

Facility: SYSTEM, System Services

Explanation: The application has attempted to read a file or device to which only write access is allowed.

User Action: If possible, correct the error in the application program.

NOSAVEDUMP, SAVEDUMP not set; dump written to PAGEFILE.SYS not saved

Facility: SDA, System Dump Analyzer

Explanation: A system dump was written to PAGEFILE.SYS because there is no SYSDUMP.DMP file. However, because the SYSGEN parameter SAVEDUMP is set to zero, the pagefile space used to hold the dump was released when the system booted; therefore, SDA cannot access the dump.

User Action: Create a SYSDUMP.DMP system dump file or set the SYSGEN parameter SAVEDUMP to 1. To make this change permanent, modify the MODPARAMS.DAT file and run AUTOGEN. For more information about making the permanent change, refer to the *OpenVMS System Manager's Manual*.

NOSUCHPAG, specified page does not exist

Facility: SYSTEM, System Services

Explanation: The specified page does not exist.

User Action: Ensure that the proper virtual address is passed to the system service.

NOT64DEVFUNC, 64-bit address not supported by device for this function

Facility: SYSTEM, System Services

Explanation: This fatal error can be returned under several conditions:

- The $QIO and $QIOW system services return this error under either of the following circumstances:
  - The caller has specified a 64-bit virtual address in the P1 device-dependent parameter but the device driver does not support 64-bit addresses with the requested I/O function.
  - The caller has specified a 64-bit address for a diagnostic buffer but the device driver does not support 64-bit addresses for diagnostic buffers.
- Some device drivers might return this condition value when 64-bit buffer addresses are passed using the P2 through P6 parameters and the driver does not support a 64-bit address with the requested I/O function.

- In the case of RMS service calls, the buffer used for the QIO request is generally an RMS intermediate buffer rather than a user buffer. There are two exceptions when the QIO transfer is made directly to or from the user buffer specified in the RAB:
  - Record I/O: RMS service $PUT to a unit record device (user's RBF buffer)
  - Block I/O: RMS service $READ (user's UBF buffer) and RMS service $WRITE (user's RBF buffer)

  In either of these cases, if a 64-bit virtual address is specified for the user's buffer but the device driver does not support 64-bit addresses with the requested I/O function, RMS returns RMS-F-SYS (QIO system service request failed) as the RMS error status (STS) in the RAB, with the system error status (NOT64DEVFUNC) returned as the status value (STV) in the RAB.

User Action: Consult the specific device driver documentation or the *OpenVMS Alpha Guide to 64-Bit Addressing* to determine which I/O functions and device drivers can support a 64-bit buffer address. If the combination of the device driver and I/O function that you are using does not permit a 64-bit buffer address, use a buffer within a 32-bit virtual address space for the $QIO and copy the data to or from the buffer that is in the 64-bit address space.

If this error is returned by an RMS service, a 32-bit virtual address should be specified in the RAB for the user buffer associated with the operation.

NOTF64, /CACHING qualifier supported only on a Files-64 disk

Facility: SET, SET Command, and SET Utility

Explanation: One of the files or directories listed on the SET FILE command is not in a Spiralog volume. You can use the /CACHING qualifier only for Spiralog files and directories.

User Action: Make sure that all the files and directories listed on the SET FILE command are in Spiralog volumes.

To check whether a volume is a Spiralog volume, use the DCL command SHOW DEVICES/FULL. The first line displayed by this command shows a device type of Files_64 for a Spiralog volume.

NOTFOUND, network 'name' was not found

Facility: NETWRK, SHOW NETWORK Command

Explanation: The specified network service is not recognized.

User Action: Check the network name and retry the operation.

NOTNOCNVRT, device does not do LBN addressing

Facility: SYSTEM, System Services

Explanation: Certain Fast-I/O operations require disk drivers to access data at the logical block number (LBN) level. This is the norm for contemporary disks. This error indicates the disk driver does not support this feature.

User Action: Move the data to be accessed to a disk that supports LBN addressing.

NOT_PROCESS_VA, specified virtual address is not a process space address

Facility: SYSTEM, System Services

Explanation: An attempt was made to modify a virtual address that is not a process space virtual address.

User Action: Ensure that the proper virtual address is passed to the system service.

NOTPERMITTED, qualifier 'qualifier-name' is not permitted

Facility: NETWRK, SET NETWORK Command

Explanation: The specified qualifier is invalid for this command.

User Action: Specify a valid qualifier and retry the operation.

NOWRTACC, write access denied

Facility: SYSTEM, System Services

Explanation: The application attempted to write a global section, file, or device to which only read access is allowed.

User Action: If possible, correct the error in the application program.

OFF_NOTBLKALGN, specified offset is not virtual disk block aligned

Facility: SYSTEM, System Services

Explanation: The specified virtual block offset is not disk block aligned.

User Action: Ensure that the virtual offset passed to the system service is disk-block aligned.

OFF_NOTPAGALGN, specified offset is not CPU-specific page aligned

Facility: SYSTEM, System Services

Explanation: The specified page offset is not CPU-specific page aligned.

User Action: Ensure that the page offset passed to the system service is CPU-specific page aligned.

OFFSET_TOO_BIG, specified offset is too large for global section being mapped

Facility: SYSTEM, System Services

Explanation: The specified offset is larger than the actual size of the global section being mapped.

User Action: Ensure that the correct section offset is passed to the system service.

OPENOUT, error opening 'file-name' as output

Facility: SDA, System Dump Analyzer

Explanation: An error occurred when SDA attempted to create the output file for a COPY operation.

User Action: Correct the problem identified in the accompanying RMS error message and retry the operation.

PAGNOTINREG, page in the specified range is not within the specified region

Facility: SYSTEM, System Services

Explanation: The specified range of pages does not exist within the specified region.

User Action: Ensure that the correct range is passed to the system service.

PAGNOTWRITE, specified page cannot be written

Facility: SYSTEM, System Services

Explanation: The specified page cannot be written.

User Action: Ensure that the correct virtual address is passed to the service.

PAGTYPVIO, operation not supported on specified page type

Facility: SYSTEM, System Services

Explanation: The attempted operation is not supported for the specified page type.

User Action: Ensure that the operation to be performed is legal for the specified page type.

PROTVIO, file protection prohibits the type of access requested

Facility: SYSTEM, System Services

Explanation: The type of access requested is not possible because of file protection.

User Action: Ensure that the correct file protection argument is passed to the system service.

QUALMISSING, required qualifier 'qualifier-name' is missing

Facility: NETWRK, SET NETWORK Command

Explanation: The qualifier specified in this message is required for this operation.

User Action: Reenter the command and include the required qualifier.

RATINGNOTINIT, LAT rating image has not been initialized

Facility: LAT, LAT Facility

Explanation: During LATACP startup, the LAT$RATING image was found to be loaded but not initialized.

User Action: Be sure to start LAT by executing the LAT$STARTUP command procedure to access the support startup files supplied by Digital. The LAT$STARTUP command procedure executes other command procedures, including LAT$CONFIG, which loads the LAT$RATING image. If you make changes to the LAT$RATING image, be sure that the initialization entry points are still called by LTDRIVER when LTDRIVER is loaded.

REGISFULL, specified region is full

Facility: SYSTEM, System Services

Explanation: The specified region is full.

User Action: If possible, delete some address space from the region, or create a new region.

REGNOTAVAIL, register not available when using System Code Debugger

Facility: SDA, System Dump Analyzer

Explanation: You cannot access the specified register when SDA is invoked from within the system code debugger.

User Action: None.

REGOWNVIO, region is owned by a more privileged access mode

Facility: SYSTEM, System Services

Explanation: The specified region is owned by a more privileged access mode.

User Action: Ensure that the correct access mode and/or region arguments are passed to the system service.

RESIGNAL_64, resignal condition to next handler

Facility: SYSTEM, System Services

Explanation: A condition handler completed without terminating or continuing the image. This message is associated with an exit status code used by 64-bit condition handling routines to indicate that the exception dispatcher continued its search for handlers.

User Action: None.

SHADZEROMBR, SHADOWING detects a zero member set

Facility: BUGCHECK, System Bugcheck

Explanation: Due to a synchronization problem, shadowing has detected a mounted virtual unit with no source members. This is an illegal state; to preserve data integrity, shadowing has crashed the node on which this condition was detected.

User Action: Document and report any events that occurred on other nodes in the VMScluster that also have the shadow set mounted. Reboot the node that crashed and remount all shadow sets.

SUPPRESSED, network-specific information suppressed due to output redirection

Facility: NETWRK, SHOW NETWORK Command

Explanation: A secondary command directed to this network service is not being executed because the /OUTPUT qualifier was specified to redirect the output.

User Action: If you wish to execute a secondary command, do not specify the /OUTPUT qualifier.

TIMEDOUT, operation timed out

Facility: NETWRK, SET/SHOW/START/STOP NETWORK Commands

Explanation: The operation timed out because another user was using the required resources.

User Action: Try the operation again later.

TIRLNGTH, object command data record length illegal 'number' record 'number' in module 'module_name' file 'file_name'

Facility: LINK, Linker Utility

Explanation: The object record length for the record is not valid. The specified length is either too large or too small for the expected data.

User Action: Contact a Digital support representative about the appropriate language processor.

TOO_MANY_ARGS, routine called with too many arguments

Facility: SYSTEM, System Services

Explanation: A system service was called with too many arguments.

User Action: Ensure that the correct number of arguments is passed to the system service.

TOOLONG, value specified is greater than 255 characters

Facility: NETWRK, SET NETWORK Command

Explanation: The value you entered exceeds 255 characters.

User Action: Enter a value consisting of 255 or fewer characters.

UNALIGNED, IOSA or data buffer not aligned

Facility: SYSTEM, System Services

Explanation: An application has supplied one of the following:

- An I/O Status Area that is not quadword aligned
- A data buffer that is not 512-byte aligned

User Action: If possible, correct the error in the application program.

VA_NOTPAGALGN, specified virtual address is not CPU-specific page aligned

Facility: SYSTEM, System Services

Explanation: The specified virtual address is not a CPU-specific, page-aligned address.

User Action: Ensure that the virtual address passed to the system service is a CPU-specific, page-aligned address.

WAIT, operation blocked by another user; please wait

Facility: NETWRK, SET/SHOW/START/STOP NETWORK Commands

Explanation: The operation cannot be performed because another user is modifying the network service information.

User Action: If you get the TIMEDOUT message, retry the operation again later.

WRITEERR, error occurred writing to output file

Facility: NETWRK, SHOW NETWORK Command

Explanation: An error was encountered while writing to the output file.

User Action: Take action based on the accompanying RMS message.

B

DEC C XPG4 Localization Utilities—OpenVMS Version 6.2 Feature

OpenVMS Version 6.2 provides XPG4-compliant utilities for managing localization data for international software applications or layered products. Localization data is defined separately from the application and is bound to it only at run time.

Note

The addition of the DEC C XPG4 code to OpenVMS Version 6.2 means that developers who link their code on OpenVMS VAX Version 6.2 will not be able to run it on OpenVMS VAX Version 6.1.

The following localization utilities are described in this appendix:

- GENCAT
- ICONV COMPILE
- ICONV CONVERT
- LOCALE COMPILE
- LOCALE LOAD
- LOCALE UNLOAD
- LOCALE SHOW CHARACTER_DEFINITIONS
- LOCALE SHOW CURRENT
- LOCALE SHOW PUBLIC
- LOCALE SHOW VALUE

These utilities are provided only on CD–ROM.

Because these utilities support the XPG4 model of internationalization, they are only useful for localizing applications written to that model. See the user documentation for each application or layered product to see if it supports XPG4 internationalization.

This appendix also describes the locale file format and the character set description (charmap) file.

B.1 GENCAT

The GENCAT command merges one or more message text source files into a message catalog file.

B.1.1 Format

GENCAT msgfile[,...] catfile

B.1.1.1 Parameters

msgfile

The file specification of a message text source file. The default file type for a message text source file is .MSGX.

catfile

The file specification of the message catalog file that is created. If *catfile* already exists, a new version is created that includes the messages in the existing catalog. The file type for a message catalog file is .CAT.

B.1.1.2 Qualifiers

None.

B.1.2 Description

The GENCAT command creates new message catalogs from one or more input source files and an existing catalog file (if there is one). A message catalog is a binary file containing the messages for an application. This includes all messages that the application issues, such as error messages, screen displays, and prompts. Applications retrieve messages from a message catalog using the `catopen`, `catgets`, and `catclose` C run-time library routines. See the *DEC C Run-Time Library Reference Manual for OpenVMS Systems* for details of these routines.

A message text source file is a text file that you create to hold messages printed by your program. You can use any text editor to enter messages into the text source file. Messages can be grouped into sets, usually to represent functional subsets of your program. Each message has a numeric identifier, which must be unique within its set. The message text source file can also contain commands recognized by GENCAT for manipulating sets and individual messages.

If a message catalog with the name *catfile* exists, GENCAT creates a new version of the file that includes the contents of the older version and then modifies it. If the catalog does not exist, GENCAT creates the catalog with the name *catfile*.

You can specify any number of message text source files. The GENCAT command processes multiple source files one after the other in the sequence that you specify them. Each successive source file modifies the catalog.

The *catfile* can contain the following commands:

message_number text

Inserts *text* as a message with the identifier *message_number*. Follow these guidelines:

- Numbers must be ascending within each set. You can skip a number, but you cannot go back to add a missing number or replace an existing number during a GENCAT session.
- If the message text is empty and a space or tab field separator is present, an empty string is stored in the message catalog.
- If a message source line has a message number but neither a field separator nor message text, the existing message with that number (if any) is deleted from the catalog.

$delset *set_number*

Deletes the set of messages indicated by *set_number*.

$quote *character*

Sets the quote character to *character*. See the Examples section for more information.

$set *set_number*

Indicates that all messages entered after this command are placed in the set indicated by *set_number*. You can change the set by entering another $set command. However, set numbers must be entered in ascending order; you cannot go back to a lower numbered set during the GENCAT session. If the command is not used, the default set number is 1.

Each initial keyword or number must be followed by white space. The GENCAT utility ignores any line that begins with a space, a tab, or a dollar sign ($) character followed by a space, a tab, or a newline character. Therefore, you can use these sequences to start comments in your *catfile*. Blank lines are also ignored. Finally, you can place comments on the same line after the $delset, $quote, or $set commands because GENCAT ignores anything that follows these commands.

A line beginning with a digit marks a message to be included in the catalog. You can specify any amount of white space between the message ID number and the message text; however, when the message text is not delimited by quotation marks, one space or tab character is recommended. When message text is not in quotation marks, GENCAT treats additional white space as part of the message. When message text is enclosed in quotation marks, GENCAT ignores all spaces or tabs between the message ID and the first quotation character.

Escape sequences like those recognized by the C language can be used in text. The escape character (\), a backslash, can be used to insert special characters in the message text. See Table B–1.

Table B–1 Special Characters

Escape Sequence	Character
\n	New Line
\t	Horizontal Tab
\v	Vertical Tab
\b	Backspace
\r	Carriage Return
\f	Form Feed
\\	Backslash Character (\ ). Use to continue message text on the following line.
ddd	The single-byte character associated with the octal value *ddd*. You can specify one, two, or three octal digits. However, you must include leading zeros if the characters following the octal digits are also valid octal digits; for example, the octal value for $ (dollar sign) is 44. To insert $5.00 into a message, use \0445.00, not \445.00; otherwise the 5 is parsed as part of the octal value.

Notes

GENCAT conforms to X/Open specifications. In an X/Open conforming application, the set numbers must be integers in the range of 1 to NL_SETMAX, inclusive; message numbers must be integers in the range of 1 to NL_MSGMAX, inclusive. NL_SETMAX and NL_MSGMAX are defined in the `<limits.h>` header file that comes with DEC C and DEC C++. For OpenVMS Version 6.2, each of these limits is 65535.

The value of LC_CTYPE from the LOCALE SHOW CURRENT command determines the interpretation of message text in the message source files *msgfile*....

B.1.3 Errors

When GENCAT reports an error, no action is taken on any commands and an existing catalog is left unchanged.

B.1.4 Examples

1.

```
$set 10 Communication Error Messages
```

This example uses the $set command in a source file to assign a set number to a group of messages.

The message set number is 10. All messages after the $set command and up to the next $set command are assigned a message set number of 10. (Set numbers must be assigned in ascending order but they need not be contiguous.) You can include a comment in the $set command.

2.

```
$delset 10 Communication Error Messages
```

This example uses the $delset command to remove from a catalog all messages belonging to the specified message set (10, in this case).

The $delset command must be placed in the proper set number order with respect to any $set commands in the same source file. You can include a comment in the $delset command.

3.

```
12 "file removed"
```

This example shows how to enter the message text and assign a message ID number to it. In this case, a message ID of 12 is assigned to the text that follows it.

You must leave at least one space or tab character between the message ID number and the message text but you can include more spaces or tabs if you prefer. If you do include more spaces or tabs, they are ignored when the message text is in quotation marks and they are considered part of the text when the message text is not in quotation marks.

Message numbers must be in ascending order within a single message set but they need not be contiguous.

All text following the message number and up to the end of the line is included as message text. If you place the escape character (\), a backslash, as the last character on the line, the message text continues on the following line. Consider the following example:

```
This is the text associated with \
message number 5.
```

The two lines in the example define the following single-line message:

```
This is the text associated with message number 5.
```

4.

```
$quote "   Use a double quote to delimit message text
$set 10            Message Facility - Quote command messages
1 "Use the $quote command to define a character \
\n for delimiting message text" \n
2 "You can include the \"quote\" character in a message \n \
by placing a \\ (backslash) in front of it" \n
3 You can include the "quote" character in a message \n \
by having another character as the first nonspace \
\n character after the message ID number \n
$quote
4 You can disable the quote mechanism by \n \
using the $quote command without \n a character \
after it \n
```

This example shows the effect of a quote character.

The $quote command defines the double quote (") as the quote character. The quote character must be the first nonspace character after the message number. Any text following the next occurrence of the quote character is ignored.

This example also shows two ways to include the quote character in the message text:

- Place a \ in front of the quote character.
- Use another character as the first nonspace character after the message number. This disables the quote character for that message only.

This example also shows the following:

- A \ is still required to split a quoted message across lines.
- To display a \ in a message, you must place another \ in front of it.
- You can format your message with a new-line character by using \n.
- If you use the $quote command with no character argument, you disable the quote mechanism.

B.2 ICONV COMPILE

The ICONV COMPILE command creates a conversion table file from a conversion source file. The conversion table file is used by the ICONV CONVERT command to convert characters in a file from one codeset to another.

B.2.1 Format

ICONV COMPILE sourcefile tablefile

B.2.1.1 Parameters

sourcefile

The file specification of the conversion source file. The default file type is .ISRC. The file naming convention Digital uses for conversion source files is:

```
fromcodeset_tocodeset.isrc
```

tablefile

The file specification of the conversion table file to be created. The default file type is .ICONV. The file naming convention for conversion table files is:

```
fromcodeset_tocodeset.iconv
```

You must follow this convention for naming conversion table files for the ICONV CONVERT command to recognize them.

Public conversion table files are in the directory defined by the logical name SYS$I18N_ICONV. Put new conversion table files in the same directory if you want to make them available systemwide.

B.2.1.2 Qualifier

/LISTING[=listfile]

Directs ICONV COMPILE to produce a listing file, which contains the source file listing and any error messages generated during compilation. If the file name is omitted from the qualifier, the default listing file name is *sourcefile*.LIS.

B.2.2 Description

The ICONV commands support any 1- to 4-byte codesets that are state independent. They do not support state-dependent codesets.

Note

There is an implementation restriction in the *tocodeset* encodings in this implementation. The characters in *tocodeset* must not use 0XFF in the fourth byte.

The conversion source file contains the character conversion rules for a specific conversion.

The format of a codeset conversion source file is defined as follows:

```
<fromcodeset_mb_cur_max>    value
<fromcodeset_mb_cur_min>    value
<tocodeset_mb_cur_max>      value
<tocodeset_mb_cur_min>      value
<fallback_code>             value
<escape_char>               value
<comment_char>              value
<fromcodeset_range>         value...value;value...value;...;value...value
ICONV_TABLE
fromvalue                         tovalue
fromvalue                         tovalue
   .                                 .
   .                                 .
   .                                 .
fromvalue                         tovalue
END ICONV_TABLE
```

where the <...> symbols and their associated values are codeset declarations, and the *fromvalue/tovalue* pairs are character conversion rules.

Codeset Declarations

The codeset declarations must precede the character conversion rules. Each declaration consists of a symbol, starting in column 1 and including the surrounding brackets, followed by one or more blanks (tabs or spaces), followed by the value to be assigned to the symbol. See Table B–2.

Table B–2 Codeset Declarations

Symbol	Value
<fromcodeset_mb_cur_max>	The maximum number of bytes in a character in the fromcodeset. This value defaults to 1.
<fromcodeset_mb_cur_min>	The minimum number of bytes in a character in the fromcodeset. This value must be less than or equal to fromcodeset_mb_cur_max. If this value is not specified, it defaults to the value of fromcodeset_mb_cur_max.
<tocodeset_mb_cur_max>	The maximum number of bytes in a character in the tocodeset. This value defaults to 1.

(continued on next page)

Table B–2 (Cont.) Codeset Declarations

Symbol	Value
<tocodeset_mb_cur_min>	The minimum number of bytes in a character in the tocodeset. This value must be less than or equal to tocodeset_mb_cur_max. If this value is not specified, it defaults to the value of tocodeset_mb_cur_max.
<fallback_code>	The *tovalue*s for the *fromvalue*s that appear in the <fromcodeset_range> but are not specified between ICONV_TABLE and END ICONV_TABLE. Specify one of three kinds of values: • SAME — specifies that the *tovalue*s are the same as the *fromvalue*s. • ERROR — specifies that the conversion from the *fromvalue* to a *tovalue* is not supported. ICONV CONVERT issues a warning and ignores the rest of the record read. The DEC C run-time library routine iconv returns to the caller with an "illegal character" error. • User-defined *tovalue* — the *fromvalue*s are converted to the specified user-defined *tovalue*. The user-defined *tovalue* can represent a multibyte character with the restriction that 0xff cannot be used as the value in the fourth byte. The settings for user-defined *tovalue*s for <fallback_code> are the same as the settings for character conversion rule values. You can use octal, decimal, or hexadecimal digits. If the <fallback_code> is not specified, it defaults to SAME.
<escape_char>	The escape character used to indicate that subsequent characters are interpreted in a special way. The escape character defaults to backslash (\).
<comment_char>	The character that, when placed in column 1 of a line, indicates that the line will be ignored. The default comment character is the number sign (#).
<fromokcodeset_range>	The fromcodeset encoding ranges. Specify this declaration if the fromcodeset is a multibyte codeset. If the fromcodeset is omitted, it defaults to a single-byte codeset and the table created by ICONV COMPILE will support only single-byte fromcodeset conversions.

When specifying codeset encoding ranges for the fromcodeset, every zone of characters must be specified. If any zones of characters are missing from the <fromcodeset_range> specification, the codeset conversion might be incorrect. It is very important to specify the codeset encoding ranges correctly for the fromcodesets supported by the rest of the DEC C run-time library (RTL). If this is not done, the codeset support for iconv and the rest of the DEC C RTL will not be consistent.

For example, the fromcodeset ranges for EUCJP are specified as:

```
<fromcodeset_range>  \x0...\x7f;\x8e\xa1...\x8e\xfe;
                     \xa1\xa1...\xfe\xfe;\x8f\xa1\xa1...\x8f\xfe\xfe
```

The settings for <fromcodeset_range> values are the same as the settings for character conversion rule values. You can use octal, decimal, or hexadecimal digits.

Character Conversion Rules

The character conversion rules are all the lines between the string ICONV_TABLE starting in column 1 and END ICONV_TABLE starting in column 1.

Character conversion rules must begin in column 1.

Empty lines and lines containing a comment_char in the first column are ignored. Comments are optional.

Character conversion rules can have one of two forms:

fromvalue *tovalue*

fromvalue...fromvalue *tovalue*

Place one or more blanks (tabs or spaces) between *fromvalue* and *tovalue*.

Use the first format to define a single-character conversion rule. For example:

```
\d32       \d101
\d37       \d106
```

Use the second format to define a range of character conversion rules. In this format, the ending *fromvalue* must be equal to or greater than the starting *fromvalue*. The subsequent *fromvalue*s defined by the range are converted to *tovalue*s in increasing order.

For example, consider the following line:

```
\d223\d32...\d223\d35       \d129\d254
```

This line is interpreted as:

```
\d223\d32       \d129\d254
\d223\d33       \d129\d255
\d223\d34       \d130\d0
\d223\d35       \d130\d1
```

For settings of *fromvalue* and *tovalue*:

- A decimal constant is defined as one, two, or three decimal digits preceded by the escape character and lowercase d. For example: \d42.
- An octal constant is defined as one, two, or three octal digits preceded by the escape character. For example: \141.
- A hexadecimal constant is defined as one or two hexadecimal digits preceded by the escape character and a lowercase x. For example: \x6a.

Each constant represents a single-byte value. You can represent multibyte values by concatenating two or more decimal, octal, or hexadecimal constants.

Note

When constants are concatenated for multibyte values, they must have the same radix (decimal, octal, or hexadecimal). Only characters in the Portable Character Set can be used to construct conversion source files.

Also see the ICONV CONVERT command.

B.2.3 Errors

If an error is encountered during processing, ICONV COMPILE does not generate an output *tablefile*. If a warning is encountered, a valid table file is created. However, because a warning can indicate a user error, you should check the returned warning messages.

Some ICONV COMPILE error messages and their descriptions follow.

```
%ICONV-E-INVFCSRNG, syntax error in <fromcodeset_range> definition
```

This error occurs when the definition of the <fromcodeset_range> symbol does not conform to the required syntax. The <fromcodeset_range> symbol defines encoding ranges and is required for multibyte codesets.

```
%ICONV-E-INVSYNTAX, invalid file syntax
```

This error occurs when a line in the source does not conform to the required syntax.

```
%ICONV-E-BADTABLE, bad table caused by invalid value for <fromcodeset_range>
definition
```

This error occurs when an invalid value is specified for the codeset encoding ranges. The encoding ranges are defined by the <fromcodeset_range> symbol.

B.2.4 Example

```
$ ICONV COMPILE/LISTING EUCTW_DECHANYU.ISRC EUCTW_DECHANYU.ICONV
```

This example shows how to create a conversion table file to convert the EUCTW codeset to the DECHANYU codeset. The listing file, EUCTW_DECHANYU.LIS, contains a listing of the source file and any error messages generated by the compiler.

B.3 ICONV CONVERT

The ICONV CONVERT command converts characters in a file from one codeset to another codeset. The converted characters are written to an output file.

B.3.1 Format

ICONV CONVERT infile outfile

B.3.1.1 Parameters

infile

The file specification of the file that contains the characters to be converted. The /FROMCODE qualifier specifies the codeset of the characters in this file.

outfile

The file specification of the file created by ICONV CONVERT. The /TOCODE qualifier specifies the codeset of the characters in this file.

B.3.1.2 Qualifiers

/FROMCODE=fromcodeset

A required qualifier that specifies the codeset of the characters in the input file *infile*.

/TOCODE=tocodeset

A required qualifier that specifies the codeset of the characters in the output file *outfile*.

B.3.2 Description

The ICONV CONVERT command converts the characters in *infile* from the codeset identified by the /FROMCODE qualifier to the codeset identified by the /TOCODE qualifier. The converted file is written to *outfile*.

The conversion is done in one of two ways:

- Using a conversion table file to look up the converted characters. This is the default method. Conversion table files are created by the DCL command ICONV COMPILE.
- Using a shareable image file that implements the required conversion. This method can be used whenever the implementation of a converter by table is either not convenient, for example, huge virtual address space versus small space by algorithm, or not possible, for example, for state dependent encoding like ISO2022.

The converter's file naming convention is:

```
fromcodeset_tocodeset.iconv
```

This naming convention is valid for both table or image file type of implementations.

Note that if you add conversion files to your system, they must use the same file-naming convention. Otherwise, the ICONV CONVERT does not recognize them.

ICONV CONVERT searches your current directory for a converter file. If it cannot find the file, it then searches the system directory defined by the logical name SYS$I18N_ICONV.

B.3.3 Example

```
$ ICONV CONVERT/FROMCODE=EUCTW/TOCODE=DECHANYU FROMFILE.DAT TOFILE.DAT
```

This example shows a conversion from EUCTW characters to DECHANYU characters. The EUCTW characters in the file FROMFILE.DAT are converted to the corresponding DECHANYU characters. The converted characters are stored in the file TOFILE.DAT.

B.4 LOCALE COMPILE

The LOCALE COMPILE command converts a locale source file into a binary locale file. The binary locale file is used by those utilities and C routines that are dependent on the setting of the international environment logical names.

B.4.1 Format

LOCALE COMPILE sourcefile

B.4.1.1 Parameter

sourcefile

The file specification of the locale source file. This file defines each category of the locale. The default file type for the source file is .LSRC. For the definition of the locale source file format, see Section B.11.

B.4.1.2 Qualifiers

/CHARACTER_DEFINITIONS=filename
/NOCHARACTER_DEFINITIONS (default)

Specifies a character-set description file (charmap) for the locale. This file maps characters to their actual character encodings. If a charmap is not specified, no symbolic names (other than collating symbols defined in a collating symbol keyword) are allowed in the locale source file. For definition of the charmap file format, see Section B.12. The default file type for a charmap is .CMAP.

/DISPLAY[=[NO]HOLE]

Used with certain Chinese locales and terminals to specify that 4-byte characters occupy four printing positions (columns) on the terminal display. The default value (/DISPLAY=NOHOLE) specifies that 4-byte characters occupy two printing positions.

/IGNORE=WARNINGS
/NOIGNORE (default)

Generates an output file even if LOCALE COMPILE issues warning messages. Use the /IGNORE keyword cautiously because the warnings could indicate user errors that you might want to correct before using the resulting locale file.

/LISTING [=filename] (batch default)
/NOLISTING (interactive default)

Specifies the name of the listing file. The /SHOW qualifier controls the information included in the listing file. If the file name is omitted, the default is *sourcefile*.LIS.

/OUTPUT=[filename]
/NOOUTPUT

Specifies the name of the output file. If the /OUTPUT qualifier is omitted, the default output file name is *sourcefile*.LOCALE. Public locales are stored in the directory defined by the logical name SYS$I18N_LOCALE. If the output file is in any other location, the locale is private.

If /NOOUTPUT is specified, the compiler does not create an output file, even if the compilation is successful.

/SHOW[=(keyword[,...])]

Use /SHOW together with /LIST to control the information included in the listing file. You can specify the following keywords:

Keyword	Description
ALL	Include all information.
BRIEF	Include a summary of the symbol table.
[NO]CHARACTER_ DEFINITIONS	Include or omit the charmap file.
NONE	Do not print any information. If NONE is specified, the listing file only contains the error messages generated.
[NO]SOURCE	Include or omit a listing of the source file.
[NO]STATISTICS	Include or omit compiler performance information.

Keyword	Description
[NO]SYMBOLS	Include or omit a listing of the charmap symbol table.
[NO]TERMINAL	Display compiler messages at the terminal.

The default is /SHOW=(SOURCE,TERMINAL).

B.4.2 Description

Use the LOCALE COMPILE command to add new locales to your system in addition to those supplied by Digital. To compile a locale, LOCALE COMPILE requires two files:

- A charmap file that defines the character set for the locale. If you do not specify a charmap file, symbolic names cannot be specified in the locale source file. If this happens, LOCALE COMPILE issues an error or warning message, depending on the category processed, and no output file is produced. (Also see the /IGNORE qualifier.)
- A locale source file. This file describes one or more of the locale categories: LC_CTYPE, LC_COLLATE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME.

B.4.3 Errors

Some LOCALE COMPILE error messages and their descriptions follow.

```
%LOCALE-E-CASEALRDY, case conversion already exists for 'character'
```

Where ‘character‘ is a character from the codeset. This error can occur when the locale compiler is processing the LC_CTYPE category. It indicates that more than one case conversion is specified for ‘character‘.

```
%LOCALE-E-PREOFCMAP, premature end of file in charmap file
```

This error occurs if there is no END CHARMAP statement in the charmap file.

```
%LOCALE-E-PREEOFSRC, premature end of file in source file
```

This error occurs if there is an error with the END statements in the locale source file.

```
%LOCALE-F-NOADDSYM, failed to add symbol to symbol table
```

This error can occur when there is insufficient memory to finish the compilation. Check the amount of memory available to your process.

```
%LOCALE-F-NOINITSYM, failed to initialize symbol table
```

This error can occur if there is insufficient memory to finish the compilation. Check the amount of memory available to your process.

B.4.4 Example

```
$ LOCALE COMPILE EN_GB_ISO8859-1/CHARACTER_DEFINITIONS=ISO8859-1 -
/LIST/SHOW=(CHARACTER_DEFINITIONS,SYMBOLS,STATISTICS)
```

This example shows how to generate a locale file named EN_GB_ISO8859-1.LOCALE from the source file EN_GB_ISO8859-1.LSRC, using the charmap file ISO8859-1.CMAP. To use this locale file, copy it to the SYS$I18N_LOCALE directory and set the LANG logical to "EN_GB.ISO8859-1". The listing file contains a listing of the charmap file, the symbol table, performance information, and any error messages generated by the compiler.

B.5 LOCALE LOAD

This command loads the specified locale name into the system's memory as shared, read-only global data.

B.5.1 Format

LOCALE LOAD name

B.5.1.1 Parameter

name

A character string that identifies the locale to be loaded. This can be one of the following:

- The name of the public locale

  Specifies the public locale. The format of the name is:

  ```
  language_country.codeset[@modifier]
  ```

  LOCALE LOAD searches for the public locale binary file in the location defined by the logical name SYS$I18N_LOCALE. The file type defaults to .LOCALE. The period (.) and at-sign (@) characters in the name specified are replaced by underscore (_) characters.

  For example, if the name specified is "zh_CN.dechanzi@radical", LOCALE LOAD searches for the following binary locale file:

  SYS$I18N_LOCALE:ZH_CN_DECHANZI_RADICAL.LOCALE

- A file specification

  Specifies the binary locale file. This can be any valid file specification. If either the device or directory is not specified, LOCALE LOAD first applies the current caller's device and directory as defaults. If the file is not found, the device and directory defined by the SYS$I18N_LOCALE logical name are used as defaults. The file type defaults to .LOCALE.

  Wildcards are not allowed. The binary locale file cannot reside on a remote node.

B.5.1.2 Qualifiers

None.

B.5.2 Description

This command loads the specified locale name into the system's memory as several shared, read-only global sections. All processes that access the loaded locale then use this one copy of the locale, thereby reducing overall demand on system memory.

LOCALE LOAD is a privileged OpenVMS command, typically issued by the system manager. The following privileges are required:

- SYSGBL
- PRMGBL

B.6 LOCALE UNLOAD

This command unloads the specified locale name from the system's memory.

B.6.1 Format

LOCALE UNLOAD name

B.6.1.1 Parameter

name

A character string that identifies the locale to be unloaded. See the LOCALE LOAD command for acceptable formats for this parameter.

B.6.1.2 Qualifiers

None.

B.6.2 Description

This command unloads the specified locale name from the system's memory. If a process is accessing the locale when the UNLOAD command is entered, the global sections are deleted after the process deaccesses the locale.

LOCALE UNLOAD is a privileged OpenVMS command, typically issued by the system manager. The following privileges are required:

- SYSGBL
- PRMGBL

Note

Only locale files loaded by the LOCALE LOAD command can be unloaded.

B.7 LOCALE SHOW CHARACTER_DEFINITIONS

This command lists character set description files (charmaps).

B.7.1 Format

LOCALE SHOW CHARACTER_DEFINITIONS

B.7.1.1 Parameters

None.

B.7.1.2 Qualifiers

None.

B.7.2 Description

This command lists all the character set description files (charmaps) in the public directory defined by the logical name SYS$I18N_LOCALE. A charmap defines the symbolic names and values of characters in a coded character set. Charmaps are used by the LOCALE COMPILE command when compiling a locale. A charmap file has the file type .CMAP.

B.7.3 Example

```
$ LOCALE SHOW CHARACTER_DEFINITIONS
[SYS$I18N.LOCALES.SYSTEM]DECHANYU
[SYS$I18N.LOCALES.SYSTEM]DECHANZI
[SYS$I18N.LOCALES.SYSTEM]DECKANJI
[SYS$I18N.LOCALES.SYSTEM]DECKOREAN
[SYS$I18N.LOCALES.SYSTEM]EUCJP
[SYS$I18N.LOCALES.SYSTEM]EUCTW
[SYS$I18N.LOCALES.SYSTEM]ISO8859-1
[SYS$I18N.LOCALES.SYSTEM]ISO8859-2
[SYS$I18N.LOCALES.SYSTEM]ISO8859-3
[SYS$I18N.LOCALES.SYSTEM]ISO8859-4
[SYS$I18N.LOCALES.SYSTEM]ISO8859-5
[SYS$I18N.LOCALES.SYSTEM]ISO8859-7
[SYS$I18N.LOCALES.SYSTEM]ISO8859-8
[SYS$I18N.LOCALES.SYSTEM]ISO8859-9
[SYS$I18N.LOCALES.SYSTEM]MITACTELEX
[SYS$I18N.LOCALES.SYSTEM]SDECKANJI
[SYS$I18N.LOCALES.SYSTEM]SJIS
```

This example shows a system with several charmap files in the SYS$I18N_ LOCALE directory.

B.8 LOCALE SHOW CURRENT

This command displays a summary of the current international environment as defined by several international environment logical names.

B.8.1 Format

LOCALE SHOW [CURRENT]

B.8.1.1 Parameters

None.

B.8.1.2 Qualifiers

None.

B.8.2 Description

The LOCALE SHOW CURRENT command lists the settings for each locale category and the values of the environment variables LC_ALL and LANG. The CURRENT keyword is the default and is, therefore, optional. The logical name that defines a category has the same name as the category. For example, the LC_MESSAGES logical name defines the setting for the LC_MESSAGES category. The locale categories are:

Category	Description
LC_COLLATE	Information about collating sequences.
LC_CTYPE	Character classification information.
LC_MESSAGES	Information about the language of program messages and the format of yes/no prompts.
LC_MONETARY	Monetary formatting information.
LC_NUMERIC	Information about formatting numbers.
LC_TIME	Time and date information.

Each locale category is defined by scanning the following logical names in the order shown, until a logical name is found. If the logical name found does not represent a valid locale file, then LOCALE SHOW displays the string "C" for all the categories.

1. LC_ALL
2. Logical names corresponding to the categories specified in the table (for example, if LC_NUMERIC is specified as a valid locale category, the LOCALE SHOW CURRENT command displays the name of the category and the locale name it defines).
3. LANG
4. SYS$LC_ALL
5. The system default for the locale categories as specified by the SYS$* logical names. For example, the default for the category LC_NUMERIC is defined by the SYS$LC_NUMERIC logical name.
6. SYS$LANG

The system manager can choose to define SYS$* logicals in the site-specific system startup files to set the default locale. If no definition is provided, programs operate using the built-in "C" locale, in which case the LOCALE SHOW CURRENT command displays the string "C" for the current locale categories.

B.8.3 Example

```
$ DEFINE LC_COLLATE EN_US.ISO8859-1 ! NOTE: the collate category in unquoted
$ DEFINE LANG EN_GB_ISO8859-1
$ DEFINE LC_MESSAGES PRIVATE$DISK:[APPL.LOCALES]SPECIAL.LOCALE
$ LOCALE SHOW CURRENT
LANG="EN_GB_ISO8859-1"
LC_CTYPE="EN_GB_ISO8859-1"
LC_COLLATE=EN_US_ISO8859-1
LC_TIME="EN_GB_ISO8859-1"
LC_NUMERIC="EN_GB_ISO8859-1"
LC_MONETARY="EN_GB_ISO8859-1"
LC_MESSAGES=PRIVATE$DISK:[APPL.LOCALES]SPECIAL.LOCALE;1
LC_ALL=
```

This example shows a process where all locale categories except LC_COLLATE and LC_MESSAGES have defaulted to the same locale, EN_GB.ISO8859-1. A setting enclosed in double quotes indicates that the setting is implied by the setting of one of the following logical names: LANG, LC_ALL, SYS$LC_ALL, or SYS$LANG. A setting not enclosed by double quotes indicates that the logical name for that category defines the international environment. This example also shows that if a locale category is specified by a complete file specification, then the complete file specification is displayed.

B.8.4 Errors

If any logical names that define the environment are improperly defined, no warning message is issued. However, the actual international environment is listed exactly as it would be seen by an application that uses the DEC C run-time library routine `setlocale` (for instance, if in the previous example the SPECIAL.LOCALE file does not exist, then the display for the LC_MESSAGES category would show LC_MESSAGES="C").

B.9 LOCALE SHOW PUBLIC

This command lists all the public locales on the system.

B.9.1 Format

LOCALE SHOW PUBLIC

B.9.1.1 Parameters

None.

B.9.1.2 Qualifiers

None.

B.9.2 Description

This command lists all the public locales on the system. The set of public locales contains all the locales that reside in the directory defined by the logical name SYS$I18N_LOCALE as well as the system's built-in locales supplied with the DEC C run-time library.

B.9.3 Example

```
$ LOCALE SHOW PUBLIC

C (Built-in)
POSIX (Built-in)
[SYS$I18N.LOCALES.SYSTEM]EN_GB_ISO8859_1
[SYS$I18N.LOCALES.SYSTEM]EN_US_ISO8859_1
[SYS$I18N.LOCALES.SYSTEM]FR_CA_ISO8859_1
[SYS$I18N.LOCALES.SYSTEM]GRBAGE_LOCALE (bad file header checksum)
[SYS$I18N.LOCALES.SYSTEM]JA_JP_DECKANJI (Permanently Loaded)
```

This example shows a system with three locale files in the SYS$I18N_LOCALE directory. The C and POSIX locales are built in with the system and, therefore, cannot be found in the SYS$I18N_LOCALE directory.

This example also shows the effect of having a bad file or a nonlocale file in the public directory and the effect of having a locale file loaded into the system's memory by the LOCALE LOAD command.

B.10 LOCALE SHOW VALUE

This command displays the value of one or more keywords from the current international environment.

B.10.1 Format

LOCALE SHOW VALUE name[,...]

B.10.1.1 Parameter

name[,...]

The name of a keyword or category. If you specify a keyword, the value of that keyword in the current locale is displayed. If you specify a category, the values of the keywords in that category are displayed. For integer keywords that have no value assigned, the value CHAR_MAX (127) is displayed. When a keyword value includes semicolons, double quotes, backslashes, or control characters, they are preceded by an escape character (usually a backslash).

Table B–3 lists the categories and keywords you can specify for *name*.

Table B–3 Locale Categories and Keywords

Category	Keyword	Keyword Description
LC_CTYPE		Character classification names
LC_TIME	DAY	Full weekday names
	ABDAY	Abbreviated weekday names
	MON	Full month names
	ABMON	Abbreviated month names
	D_T_FMT	Date and time format
	D_FMT	Date format
	T_FMT	Time format
	T_FMT_AMPM	Time format in the 12-hour clock
	AM_PM	Defines how the ante meridiem (a.m.) and post meridiem (p.m.) strings are represented
	ERA	Defines how years are counted and displayed for eras in a locale
	ERA_D_FMT	Era date format
	ERA_D_T_FMT	Era date and time format
	ERA_T_FMT	Era time format
	ALT_DIGITS	String defining alternative symbols for digits
LC_NUMERIC	DECIMAL_POINT	Character used as a decimal delimiter
	THOUSANDS_SEP	Character used to group digits to the left of the decimal delimiter
	GROUPING	Defines how characters to the left of the decimal delimiter are grouped
LC_MONETARY	INT_CURR_SYMBOL	Character string representing the international currency symbol
	CURRENCY_SYMBOL	String used as the local currency symbol
	MON_DECIMAL_POINT	Character used as a decimal delimiter when formatting monetary quantities
	MON_THOUSANDS_SEP	Character used as a separator for groups of digits to the left of the decimal delimiter
	POSITIVE_SIGN	String used to represent positive monetary quantities

(continued on next page)

Table B–3 (Cont.) Locale Categories and Keywords

Category	Keyword	Keyword Description
	NEGATIVE_SIGN	String used to represent negative monetary quantities
	INT_FRAC_DIGITS	Number of digits displayed to the right of the decimal delimiter when formatting monetary quantities using the international currency symbol
	FRAC_DIGITS	Number of digits displayed to the right of the decimal delimiter when formatting monetary quantities using the local currency symbol
	P_CS_PRECEDES	For positive monetary values, this is set to 1 if the local currency symbol precedes the number and 0 if the symbol follows the number
	N_CS_PRECEDES	For negative monetary values, this is set to 1 if the local currency symbol precedes the number and 0 if the symbol follows the number
	P_SEP_BY_SPACE	For positive monetary values, this is set to 0 if there is no space between the currency symbol and the value, 1 if there is a space, and 2 if there is a space between the symbol and the sign string
	N_SEP_BY_SPACE	For negative monetary values, this is set to 0 if there is no space between the currency symbol and the value, 1 if there is a space, and 2 if there is a space between the symbol and the sign string
	P_SIGN_POSN	Integer used to indicate where the POSITIVE_SIGN string should be placed
	N_SIGN_POSN	Integer used to indicate where the NEGATIVE_SIGN string should be placed
	MON_GROUPING	Defines how digits are grouped when formatting monetary values
LC_MESSAGES	YESSTR	String representing YES in the current locale
	NOSTR	String representing NO in the current locale
	YESEXPR	Expression representing an affirmative response in the current locale
	NOEXPR	Expression representing a negative response in the current locale

Note

When an environment variable that affects the setting of the current locale points to an invalid locale, then the "C" locale is set.

Other valid keywords that are not displayed by default as part of any category include:

- CHARMAP — displays the file specification of the charmap used when the locale was created.
- CODE_SET_NAME — defines the name of the coded character set for which the charmap file is defined.

- MB_CUR_MAX — defines the maximum number of bytes in a multibyte character.
- MB_CUR_MIN — defines the minimum number of bytes in a character in the coded character set.

B.10.1.2 Qualifiers

/CATEGORY

Displays the category name before each keyword. If /CATEGORY is not specified, the category name is not displayed.

/KEYWORD

Displays the keyword name before the value of a keyword. If /KEYWORD is not specified, the value of the keyword is displayed, but not its name.

B.10.2 Errors

```
%LOCALE-E-NOKEYFND, no keyword keyword-name found
```

The *keyword-name* is not a valid keyword. Specify only the keywords listed in Table B–3.

B.10.3 Description

This command displays the value of one or more keywords from the current international environment.

B.10.4 Examples

1.

```
$ LOCALE SHOW VALUE NOEXPR
"^[nN][[:alpha:]]*"
```

Issuing LOCALE SHOW VALUE without qualifiers displays the value of the NOEXPR string.

2.

```
$ LOCALE SHOW VALUE/CATEGORY NOEXPR
LC_MESSAGES
"^[nN][[:alpha:]]*"
```

Specifying /CATEGORY displays the category name (LC_MESSAGES) before the value of the NOEXPR string.

3.

```
$ LOCALE SHOW VALUE/KEYWORD NOEXPR
noexpr= "^[nN][[:alpha:]]*"
```

Specifying /KEYWORD displays the keyword name before its value.

4.

```
$ LOCALE SHOW VALUE/KEYWORD/CATEGORY NOEXPR
LC_MESSAGES
noexpr= "^[nN][[:alpha:]]*"
```

Specifying /KEYWORD and /CATEGORY displays the category and keyword name before the keyword value.

B.11 Locale File Format

A locale definition source file contains one or more categories that describe a locale. You can convert a locale definition source file into a locale by using the LOCALE COMPILE command. Locales can be modified only by editing a locale definition source file and then using the LOCALE COMPILE command again on the new source file. Each locale source file section defines a category of locale data. A source file cannot contain more than one section for the same category.

B.11.1 Locale Categories

The following standard locale categories are supported:

- LC_COLLATE — Defines character or string collation information
- LC_CTYPE — Defines character classification, case conversion, and other character attributes
- LC_MESSAGES — Defines the format for affirmative and negative responses
- LC_MONETARY — Defines rules and symbols for formatting monetary numeric information
- LC_NUMERIC — Defines a list of rules and symbols for formatting nonmonetary numeric information
- LC_TIME — Defines a list of rules and symbols for formatting time and date information

B.11.1.1 Overriding Defaults

You can include optional declarations at the beginning of your locale source file to override the default comment and escape characters used in locale category definitions:

- Escape character

  The escape character is used in decimal or hexadecimal constants when they are specified in the locale file. The default escape character is the backslash (\). To define another escape character, include a line with the following format:

  ```
  escape_char  <char_symbol>
  ```

- Comment character

  The comment character is the first character of any comment entries in the locale file. The default comment character is the number sign (#). To define another comment character, use the following format:

  ```
  comment_char  <char_symbol>
  ```

In the preceding formats, <*char_symbol*> is the character's symbolic name as defined in the charmap file used to build the locale's codeset. One or more blank characters (spaces or tabs) must separate `escape_char` or `comment_char` from <*char_symbol*>.

B.11.1.2 Category Source Definitions

Each category source definition consists of the following:

- The category header (*category_name*)
- The associated keyword or value pairs that comprise the category body
- The category trailer (END *category_name*)

For example:

```
LC_CTYPE
<source for LC_CTYPE category>
END LC_CTYPE
```

The source for all of the categories is specified using keywords, strings, character literals, and character symbols. Each keyword identifies either a definition or a rule. The remainder of the statement containing the keyword contains the operands to the keyword. Operands are separated from the keyword by one or more blank characters (spaces or tabs). A statement may be continued on the next line by placing a backslash (\) as the last character before the new-line character that terminates the line. Lines containing the comment character (#) in the first column are treated as comment lines.

A symbolic name begins with the left angle-bracket character (<) and ends with the right angle-bracket character (>). The characters between the < and the > can be any characters from the Portable Character Set, except for the control and space characters. For example, <A-diaeresis> could be a symbolic name for a character. Any symbolic name referenced in the locale source file must be defined via the Portable Character Set or in the character set description (charmap) file for that locale.

A character literal is the character itself, or a decimal, hexadecimal, or octal constant. A decimal constant contains two or three decimal digits and has the following form, where *n* is any decimal digit:

\d*nn* or \d*nnn*

A hexadecimal constant contains two hexadecimal digits and has the following form, where *n* is any hexadecimal digit:

\x*nn*

An octal constant contains two or three octal digits and has the following form, where *n* is any octal digit:

nn or *nnn*

The explicit definition of each category in a locale definition source file is not required. When a category is undefined in a locale definition source file, the LOCALE COMPILE command will not store any data value for this category in the resulting locale file.

B.11.2 LC_COLLATE Category

The LC_COLLATE category defines the relative order between collation items. This category begins with the LC_COLLATE header and ends with the END LC_COLLATE trailer.

A collation item is the unit of comparison for collation. A collation item may be a character or a sequence of characters. Every collation item in the locale has a set of weights, which determine if the collation item collates before, equal to, or after the other collation items in the locale. Each collation item is assigned collation weights by the LOCALE COMPILE command when the locale definition source file is compiled. These collation weights are then used by applications programs that compare strings.

String comparison is performed by comparing the collation weights of each character in the string until either a difference is found or the strings are determined to be equal. This comparison may be performed several times if the locale defines multiple collation orders. For example, in the French locale, the strings are compared using a primary set of collation weights. If they are equal on the basis of this comparison, they are compared again using a secondary set of collation weights. A collation item has a set of collation weights associated with it that is equal to the number of collation sort rules defined for the locale.

Every character defined in the charmap file (or every character in the Portable Character Set if no charmap file is specified) is itself a collation item. Additional collation items can be defined using the `collating-element` statement (see the description that follows).

Table B–4 lists the statement keywords recognized in the LC_COLLATE category.

Table B–4 LC_COLLATE Category Keywords

Keyword	Description
`copy`	Specifies the name of an existing locale to be used as the definition of this category. If you specify a copy statement, you need not specify any other keywords in this category.
`collating-element`	Specifies multicharacter collation items.
`collating-symbol`	Specifies collation symbols for use in collation sequence statements.
`order_start`	Specifies collation order statements that assign collation weights to collation items.

The `collating-element`, `collating-symbol`, and `order_start` statements are further described in the following sections.

B.11.2.1 The collating-element Statement

The `collating-element` statement specifies multicharacter collation items.

Syntax:

`collating-element <`*`character_symbol`*`> from <`*`string`*`>`

The *character_symbol* argument defines a collation item that is a string of one or more characters as a single collation item. The *character_symbol* cannot duplicate any symbolic name in the current charmap file or any other symbolic name defined in this collation definition.

The *string* argument specifies a string of two or more characters that define the *character_symbol* argument. The following are examples of the syntax for the collating-element statement:

```
collating-element <ch> from "<c><h>"
collating-element <e-acute> from "<acute><e>"
collating-element <11> from "<1><1>"
```

A *character_symbol* argument defined by the collating-element statement is recognized only within the LC_COLLATE category.

B.11.2.2 The collating-symbol Statement

The collating-symbol statement specifies collation symbols for use in collation sequence statements.

Syntax:

```
collating-symbol <collating_symbol>
```

The collating_symbol argument cannot duplicate any symbolic name in the current charmap file or any other symbolic name defined in this collation definition. The following are examples of collating-symbol statements:

```
collating-symbol <UPPER_CASE>
collating-symbol <HIGH>
```

An argument defined by the collating_symbol statement is recognized only within the LC_COLLATE category.

B.11.2.3 The order_start Statement

The order_start statement is followed by one or more collation order statements that assign collation weights to collation items and the order_end keyword. The order_start statement is a required statement.

Syntax:

```
order_start sort_rules;sort_rules;...;sort_rules
collation_order_statements
order_end
```

Sort Rules

The *sort_rules* directives have the following syntax:

```
keyword, keyword,...,keyword
```

where *keyword* is FORWARD, BACKWARD, or POSITION.

The *sort_rules* directives are optional. If specified, they define the rules to apply during string comparison. The number of specified *sort_rules* directives defines the number of weights each collation item is assigned (that is, the directives define the number of collation orders in the locale). If no *sort_rules* directives are specified, one forward directive is assumed and comparisons are made on a character basis rather than a string basis.

If *sort_rules* directives are present, the first one applies when comparing strings that use the primary weight, the second when comparing strings that use the secondary weight, and so on. Each set of *sort_rules* directives is separated by a semicolon (;). A *sort_rules* directive consists of one or more keywords separated by commas. The following keywords are supported:

FORWARD — Specifies that collation weight comparisons proceed from the beginning of a string to the end of the string.

BACKWARD — Specifies that collation weight comparisons proceed from the end of a string to the beginning of the string.

POSITION — Specifies that collation weight comparisons consider the relative position of nonignored elements in the string (that is, if strings compare as equal, the element with the shortest distance from the starting point of the comparison collates first).

The forward and backward keywords are mutually exclusive.

Here is an example of a *sort_rules* directive:

```
order_start         forward;backward
```

Collation Order Statements

The following syntax rules apply to the collation order statements:

- Each collation order statement consists of a <*character_symbol*> specification followed by white space and a set of collation orders.
- Characters in the character set can be explicitly specified in the collation order statements or implicitly specified using the ellipsis symbol (...).
- A collation order statement that begins with the UNDEFINED special symbol specifies any characters that are in the character set but not explicitly or implicitly specified by other collation order statements.

The optional operands for each collation item are used to define the primary, secondary, or subsequent weights for the collation item. The special symbol IGNORE is used to indicate a collation item that is to be ignored when strings are compared.

An ellipsis keyword appearing in place of a collating_element_list indicates the weights are to be assigned, for the characters in the identified range, in numerically increasing order from the weight for the character symbol on the left side of the preceding statement.

The use of the ellipsis keyword results in a locale that may collate differently when compiled with different character set description (charmap) source files.

The UNDEFINED special symbol includes all coded character set values not specified explicitly or with an ellipsis symbol. These characters are inserted in the character collation order at the point indicated by the UNDEFINED special symbol and are all assigned the same weight. If no UNDEFINED special symbol exists and the collation order does not specify all collation items from the coded character set, a warning is issued and all undefined characters are placed at the end of the character collation order.

Example

The following is an example of a collation order statement section in the LC_COLLATE locale definition source file category:

```
order_start     forward;backward
UNDEFINED       IGNORE;IGNORE
<LOW>
<space>         <LOW>;<space>
...              <LOW>;...
<a>             <a>;<a>
<a-acute>       <a>;<a-acute>
<a-grave>       <a>;<a-grave>
<A>             <a>;<A>
<A-acute>       <a>;<A-acute>
<A-grave>       <a>;<A-grave>
<ch>            <ch>;<ch>
<Ch>            <ch>;<Ch>
<s>             <s>;<s>
<ss>            <s><s>;<s><s>
<eszet>         <s><s>;<eszet><eszet>
...             <HIGH>;...
<HIGH>
order_end
```

This example is interpreted as follows:

- The UNDEFINED special symbol indicates that all characters not specified in the definition (either explicitly or by the ellipsis symbol) are ignored for collation purposes.
- All collation items between <space> and <a> have the same primary equivalence class and individual secondary weights based on their coded character-set values.
- All versions of the letter a (uppercase and lowercase, and with or without diacriticals) belong to the same primary collation class.
- The <c><h> multicharacter collation item is represented by the <ch> collating symbol and belongs to the same primary equivalence class as the <C><h> multicharacter collation item.
- The <eszet> character is collated as an <s><s> string (that is, one <eszet> character is expanded to two characters before comparing).

B.11.3 LC_CTYPE Category

The LC_CTYPE category defines character classification, case conversion, and other character attributes. This category begins with the LC_CTYPE header and ends with the END LC_CTYPE trailer.

All operands for LC_CTYPE category statements are defined as lists of characters. Each list consists of one or more characters or symbolic character names separated by semicolons. An ellipsis (...) can represent a series of characters; for example, <a>;...;<z> represents the characters in the range a through z.

Table B–5 lists the statement keywords recognized in the LC_CTYPE category. In the keyword descriptions, the phrase "automatically included" means that an error does not occur if the referenced characters are included or omitted; the characters are provided if they are missing, and are accepted if they are present.

Table B–5 LC_CTYPE Category Keywords

Keyword	Description
copy	Specifies the name of an existing locale to be used as the definition for this category. If you specify a copy statement, you cannot specify any other keyword.
upper	Defines uppercase letter characters. Do not specify any character defined by the cntrl, digit, punct, or space keyword. The uppercase letters A through Z are automatically included in this set.
lower	Defines lowercase letter characters. Do not specify any character defined by the cntrl, digit, punct, or space keyword. The lowercase letters a through z are automatically included in this set.
alpha	Defines all letter characters. Do not specify any character defined by the cntrl, digit, punct, or space keyword. Characters defined by the upper and lower keywords are automatically included in this character class.
digit	Defines numeric digit characters. Only the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 can be specified. The digits 0 through 9 are automatically included in this set.
space	Defines white-space characters. Do not specify any character defined by the upper, lower, alpha, digit, graph, or xdigit keyword. The space, form-feed, new-line, carriage-return, tab, and vertical tab characters are automatically included in this set.
cntrl	Defines control characters. Do not specify any character defined by the upper, lower, alpha, digit, punct, graph, print, or xdigit keyword.
punct	Defines punctuation characters. Do not specify the space character or any character defined by the upper, lower, alpha, digit, cntrl, or xdigit keywords.
graph	Defines printable characters, excluding the space character. Do not specify any character defined by the cntrl keyword. The characters defined by the upper, lower, alpha, digit, xdigit, and punct keywords are automatically included in this character class.
print	Defines printable characters, including the space character. Do not specify any character defined by the cntrl keyword. The space character and characters defined by the upper, lower, alpha, digit, xdigit, and punct keywords are automatically included in this character class.
xdigit	Defines hexadecimal digit characters. Only the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 can be specified. Any character, however, can be specified for the hexadecimal values for 10 to 15. These alternate hexadecimal digits are not used by standard conversion routines when converting digit strings from hexadecimal to numeric quantities. The numbers 0 through 9 and the letters A through F and a through f are automatically included in this set.

(continued on next page)

Table B–5 (Cont.) LC_CTYPE Category Keywords

Keyword	Description
`blank`	Defines blank characters. The space and horizontal tab characters are included in this character class. Any characters defined by this statement are automatically included in the `space` class.
`toupper`	Defines the mapping of lowercase characters to uppercase characters. Operands for this keyword consist of character pairs separated by commas. Each character pair is enclosed in parentheses ( ) and separated from the next pair by a semicolon (;). The first character in each pair is considered a lowercase character; the second character is considered an uppercase character. Only characters defined by the `lower` and `upper` keywords can be specified. If `toupper` is not specified, a through z is mapped to A through Z by default.
`tolower`	Defines the mapping of uppercase characters to lowercase characters. Operands for this keyword consist of character pairs separated by commas. Each character pair is enclosed in parentheses ( ) and separated from the next pair by a semicolon (;). The first character in each pair is considered an uppercase character; the second character is considered a lowercase character. Only characters defined by the `lower` and `upper` keywords can be specified. If `tolower` is not specified, the mapping defaults to the reverse mapping of the `toupper` keyword, if specified. If the `toupper` and `tolower` keywords are both omitted, the mapping for each defaults to that of the C locale.

Additional keywords can be provided to define new character classifications. For example:

```
charclass vowel
vowel        <a>;<e>;<i>;<o>;<u>;<y>
```

The LC_CTYPE category does not support multicharacter elements (for example, the German Eszet character is traditionally classified as a lowercase letter). In proper capitalization of German text, the Eszet character is replaced by the two characters SS; there is no corresponding uppercase letter. This kind of conversion is outside the scope of the `toupper` and `tolower` keywords.

The following is an example of a possible LC_CTYPE category listed in a locale definition source file:

```
LC_CTYPE
#"alpha" is by default "upper" and "lower"
#"alnum" is by definition "alpha" and "digit"
#"print" is by default "alnum", "punct" and the space character
#"graph" is by default "alnum" and "punct"
#"tolower" is by default the reverse mapping of "toupper"
#
upper   <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
        <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
#
lower   <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
        <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>
#
digit   <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
        <seven>;<eight>;<nine>
#
space   <tab>;<newline>;<vertical-tab>;<form-feed>;\
        <carriage-return>;<space>
#
cntrl   <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
        <form-feed>;<carriage-return>;<NUL>;<SOH>;<STX>;\
        <ETX>;<EOT>;<ENQ>;<ACK>;<SO>;<SI>;<DLE>;<DC1>;<DC2>;\
        <DC3>;<DC4>;<NAK>;<SYN>;<ETB>;<CAN>;<EM>;<SUB>;\
        <ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>
#
punct   <exclamation-mark>;<quotation-mark>;<number-sign>;\
        <dollar-sign>;<percent-sign>;<ampersand>;<asterisk>;\
        <apostrophe>;<left-parenthesis>;<right-parenthesis>;\
        <plus-sign>;<comma>;<hyphen>;<period>;<slash>;\
        <colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
        <greater-than-sign>;<question-mark>;<commercial-at>;\
        <left-square-bracket>;<backslash>;<circumflex>;\
        <right-square-bracket>;<underline>;<grave-accent>;\
        <left-curly-bracket>;<vertical-line>;<tilde>;\
        <right-curly-bracket>
#
xdigit  <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
        <seven>;<eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;\
        <a>;<b>;<c>;<d>;<e>;<f>
#
blank   <space>;<tab>
#
toupper (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
        (<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
        (<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
        (<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
        (<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
        (<z>,<Z>)
#
END LC_CTYPE
```

B.11.4 LC_MESSAGES Category

The LC_MESSAGES category defines the format for affirmative and negative system responses. This category begins with the LC_MESSAGES header and ends with the END LC_MESSAGES trailer.

All operands for the LC_MESSAGES category are defined as strings or extended regular expressions bounded by double quotation marks ("). These operands are separated from the keyword they define by one or more blank characters (spaces or tabs). Two adjacent double quotation marks ("") indicate an undefined value.

Table B–6 lists the statement keywords recognized in the LC_MESSAGES category.

Table B–6 LC_MESSAGES Category Keywords

Keyword	Description
copy	Specifies the name of an existing locale to be used as the definition of this category. If you specify a copy statement, you cannot specify any other keyword.
yesexpr	Specifies an extended regular expression that describes the acceptable affirmative response to a question expecting an affirmative or negative response.
noexpr	Specifies an extended regular expression that describes the acceptable negative response to a question expecting an affirmative or negative response.
yesstr	Specifies the locale's equivalent of an acceptable affirmative response. This string is accessible to applications through the nl_langinfo subroutine as nl_langinfo (YESSTR). Note that yesstr is likely to be withdrawn from the XPG4 standard; yesexpr is the recommended alternative.
nostr	Specifies the locale's equivalent of an acceptable negative response. This string is accessible to applications through the nl_langinfo subroutine as nl_langinfo (NOSTR). Note that nostr is likely to be withdrawn from the XPG4 standard; noexpr is the recommended alternative.

The following is an example of a possible LC_MESSAGES category listed in a locale definition source file:

```
LC_MESSAGES
#
yesexpr "<circumflex><left-square-bracket><y><Y>\
<right-square-bracket>"
noexpr  "<circumflex><left-square-bracket><n><N>\
<right-square-bracket>"
yesstr  "<y><e><s>"
nostr   "<n><o>"
#
END LC_MESSAGES
```

B.11.5 LC_MONETARY Category

The LC_MONETARY category defines rules and symbols for formatting monetary numeric information. This category begins with the LC_MONETARY header and ends with an END LC_MONETARY trailer.

B.11.5.1 LC_MONETARY Keywords

All operands for the LC_MONETARY category keywords are defined as string or integer values. String values are bounded by double quotation marks ("). All values are separated from the keyword they define by one or more blank characters (spaces or tabs). Two adjacent double quotation marks ("") indicate an undefined string value. A negative one (–1) indicates an undefined integer value.

Table B–7 lists the statement keywords recognized in the LC_MONETARY category.

Table B–7 LC_MONETARY Category Keywords

Keyword	Description
copy	Specifies the name of an existing locale to be used as the definition of this category. If you specify a copy statement, you cannot specify any other keyword.
int_curr_symbol	Specifies the string used for the international currency symbol. The operand for this keyword is a 4-character string†. The first three characters contain the alphabetic international currency symbol. The fourth character defines a character separator for insertion between the international currency symbol and a monetary quantity.
currency_symbol	Specifies the string used for the local currency symbol.
mon_decimal_point	Specifies the decimal delimiter string used for formatting monetary quantities.
mon_thousands_sep	Specifies the character separator used for grouping digits to the left of the decimal delimiter in formatted monetary quantities.
mon_grouping	Specifies a string that defines the size of each group of digits in formatted monetary quantities. The operand for this keyword consists of a sequence of integers separated by semicolons. Each integer specifies the number of digits in a group. The first integer defines the size of the group immediately to the left of the decimal delimiter. Subsequent integers define succeeding groups to the left of the previous group. If the last integer is not –1, it is used to group any remaining digits. If the last integer is –1, no further grouping is performed. A sample interpretation of the mon_grouping statement follows. Assuming a value of 123456789 to be formatted and a mon_thousands_sep operand of ' (single quotation mark), the following results occur: mon_grouping Formatted Value 3;-1 123456'789 3 123'456'789 3;2;-1 1234'56'789 3;2 12'34'56'789
positive_sign	Specifies the string used to indicate a nonnegative formatted monetary quantity.
negative_sign	Specifies the string used to indicate a negative formatted monetary quantity.
int_frac_digits	Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the int_curr_symbol value.

†The current implementation of DEC C RTL allows more than four characters to be specified. However, the user should not rely on this fact and use it exactly as specified. The 4-character limit will be implemented in a future version of DEC C RTL.

(continued on next page)

Table B–7 (Cont.) LC_MONETARY Category Keywords

Keyword	Description
frac_digits	Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the currency_symbol value.
p_cs_precedes	Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string precedes or follows the value for a nonnegative-formatted monetary quantity. The following integer values are recognized: 0 The currency symbol follows the monetary quantity. 1 The currency symbol precedes the monetary quantity.
p_sep_by_space	Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated by a space from a nonnegative-formatted monetary quantity. The following integer values are recognized: 0 No space separates the currency symbol from the monetary quantity. 1 A space separates the currency symbol from the monetary quantity. 2 A space separates the currency symbol and the positive_sign string, if adjacent.
n_cs_precedes	Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string precedes or follows the value for a negative-formatted monetary quantity. The following integer values are recognized: 0 The currency symbol follows the monetary quantity. 1 The currency symbol precedes the monetary quantity.
n_sep_by_space	Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated by a space from a negative-formatted monetary quantity. The following integer values are recognized: 0 No space separates the currency symbol from the monetary quantity. 1 A space separates the currency symbol from the monetary quantity. 2 A space separates the currency symbol and the negative_sign string, if adjacent.

(continued on next page)

Table B–7 (Cont.) LC_MONETARY Category Keywords

Keyword	Description
p_sign_posn	Specifies an integer value indicating the positioning of the positive_sign string for a nonnegative-formatted monetary quantity. The following integer values are recognized: 0 A left_parenthesis and right_parenthesis symbol enclose both the monetary quantity and the int_curr_symbol or currency_symbol string. 1 The positive_sign string precedes the quantity and the int_curr_symbol or currency_symbol string. 2 The positive_sign string follows the quantity and the int_curr_symbol or currency_symbol string. 3 The positive_sign string immediately precedes the int_curr_symbol or currency_symbol string. 4 The positive_sign string immediately follows the int_curr_symbol or currency_symbol string.
n_sign_posn	Specifies an integer value indicating the positioning of the negative_sign string for a negative-formatted monetary quantity. The following integer values are recognized: 0 A left_parenthesis and right_parenthesis symbol enclose both the monetary quantity and the int_curr_symbol or currency_symbol string. 1 The negative_sign string precedes the quantity and the int_curr_symbol or currency_symbol string. 2 The negative_sign string follows the quantity and the int_curr_symbol or currency_symbol string. 3 The negative_sign string immediately precedes the int_curr_symbol or currency_symbol string. 4 The negative_sign string immediately follows the int_curr_symbol or currency_symbol string.

B.11.5.2 Monetary Format Variations

You can produce a unique customized monetary format by changing the value of a single statement. Table B–8 shows the results of using all combinations of defined values for the p_cs_precedes, p_sep_by_space, and p_sign_posn statements.

Table B–8 Monetary Format Variations

	p_sep_by_space =	2	1	0
p_cs_precedes = 1	p_sign_posn = 0	($1.25)	($ 1.25)	($1.25)
	p_sign_posn = 1	+ $1.25	+$ 1.25	+$1.25
	p_sign_posn = 2	$1.25 +	$ 1.25+	$1.25+
	p_sign_posn = 3	+ $1.25	+$ 1.25	+$1.25
	p_sign_posn = 4	$ +1.25	$+ 1.25	$+1.25

(continued on next page)

Table B–8 (Cont.) Monetary Format Variations

	p_sep_by_space =	2	1	0
p_cs_precedes = 0	p_sign_posn = 0	(1.25 $)	(1.25 $)	(1.25$)
	p_sign_posn = 1	+1.25 $	+1.25 $	+1.25$
	p_sign_posn = 2	1.25$ +	1.25 $+	1.25$+
	p_sign_posn = 3	1.25+ $	1.25 +$	1.25+$
	p_sign_posn = 4	1.25$ +	1.25 $+	1.25$+

The following is a sample LC_MONETARY category specified in a locale definition source file:

```
LC_MONETARY
#
int_curr_symbol         "<U><S><D><space>"
currency_symbol         "<dollar-sign>"
mon_decimal_point       "<period>"
mon_thousands_sep       "<comma>"
mon_grouping            3
positive_sign           "<plus-sign>"
negative_sign           "<hyphen>"
int_frac_digits         2
frac_digits             2
p_cs_precedes           1
p_sep_by_space          2
n_cs_precedes           1
n_sep_by_space          2
p_sign_posn             3
n_sign_posn             3
#
END LC_MONETARY
```

B.11.6 LC_NUMERIC Category

The LC_NUMERIC category defines rules and symbols for formatting nonmonetary numeric information. This category begins with the LC_NUMERIC header and ends with the END LC_NUMERIC trailer.

All operands for the LC_NUMERIC category keywords are defined as string or integer values. String values are bounded by double quotation marks ("). All values are separated from the keyword they define by one or more blank characters (spaces or tabs). Two adjacent double quotation characters ("") indicate an undefined string value. A negative one (–1) indicates an undefined integer value.

Table B–9 lists the statement keywords recognized in the LC_NUMERIC category.

Table B–9 LC_NUMERIC Category Keywords

Keyword	Description
copy	Specifies the name of an existing locale to be used as the definition of this category. If you specify a copy statement, you cannot specify any other keyword.
decimal_point	Specifies the decimal delimiter string used to format nonmonetary numeric quantities. This keyword cannot be omitted and cannot be set to the undefined string value.
thousands_sep	Specifies the string separator used for grouping digits to the left of the decimal delimiter in formatted nonmonetary numeric quantities.
grouping	Defines the size of each group of digits in formatted monetary quantities. The operand for the grouping keyword consists of a sequence of integers separated by semicolons. Each integer specifies the number of digits in a group. The first integer defines the size of the group immediately to the left of the decimal delimiter. Subsequent integers define succeeding groups to the left of the previous group. Grouping is performed for each integer specified for the grouping keyword. If the last integer is not –1, it is used repeatedly to group any remaining digits. If the last integer is –1, no more grouping is performed. A sample interpretation of the grouping statement follows. Assuming a value of 123456789 to be formatted and a thousands_sep operand of ' (single quotation mark), the following results occur: grouping Formatted Value 3;-1 123456'789 3 123'456'789 3;2;-1 1234'56'789 3;2 12'34'56'789

The following is a sample LC_NUMERIC category specified in a locale definition source file:

```
LC_NUMERIC
#
decimal_point   "<period>"
thousands_sep   "<comma>"
grouping        <3>
#
END LC_NUMERIC
```

B.11.7 LC_TIME Category

The LC_TIME category defines rules and symbols for formatting time and date information. This category begins with the LC_TIME category header and ends with the END LC_TIME trailer.

All operands for the LC_TIME category keywords are defined as string or integer values. String values are bounded by double quotation marks ("). All values are separated from the keyword they define by one or more blank characters (spaces or tabs). Two adjacent double quotation characters ("") indicate an undefined string value. Field descriptors, described later in this section, are used by commands and subroutines that query the LC_TIME category to represent elements of time and date formats.

B.11.7.1 Keywords

Table B–10 lists the statement keywords recognized in the LC_TIME category.

Table B–10 LC_TIME Category Keywords

Keyword	Description
`copy`	Specifies the name of an existing locale to be used as the definition of this category.
	If you specify a `copy` statement, you cannot specify any other keyword.
`abday`	Defines the abbreviated weekday names corresponding to the %a field descriptor.
	Recognized values consist of seven strings separated by semicolons. The first string corresponds to the abbreviated name for the first day of the week (Sun), the second to the abbreviated name for the second day of the week, and so on.
`day`	Defines the full spelling of the weekday names corresponding to the %A field descriptor.
	Recognized values consist of seven strings separated by semicolons. The first string corresponds to the full spelling of the name of the first day of the week (Sunday), the second to the name of the second day of the week, and so on.
`abmon`	Defines the abbreviated month names corresponding to the %b field descriptor.
	Recognized values consist of 12 strings separated by semicolons. The first string corresponds to the abbreviated name for the first month of the year (Jan), the second to the abbreviated name for the second month of the year, and so on.
`mon`	Defines the full spelling of the month names corresponding to the %B field descriptor.
	Recognized values consist of 12 strings separated by semicolons. The first string corresponds to the full spelling of the name for the first month of the year (January), the second to the full spelling of the name for the second month of the year, and so on.
`d_t_fmt`	Defines the string used for the standard date-and-time format corresponding to the %c field descriptor. The string can contain any combination of characters and field descriptors.
`d_fmt`	Defines the string used for the standard date format corresponding to the %x field descriptor. The string can contain any combination of characters and field descriptors.
`t_fmt`	Defines the string used for the standard time format corresponding to the %X field descriptor. The string can contain any combination of characters and field descriptors.

(continued on next page)

Table B–10 (Cont.) LC_TIME Category Keywords

Keyword	Description
am_pm	Defines the strings used to represent a.m. (before noon) and p.m. (afternoon) corresponding to the %p field descriptor. Recognized values consist of two strings separated by semicolons. The first string corresponds to the a.m. designation, the second string corresponds to the p.m. designation.
t_fmt_ampm	Defines the string used for the standard 12-hour time format that includes an am_pm value (%p field descriptor). This statement corresponds to the %r field descriptor. The string can contain any combination of characters and field descriptors. If the string is empty, the 12-hour format is not supported by the locale.
era	Defines how the years are counted and displayed for each era in a locale, corresponding to the %E field descriptor modifier. For each era, there must be one string in the following format: *direction*:*offset*:*start_date*:*end_date*:*name*:*format* The variables for the era string format are defined as follows: • *direction* — Specifies a minus (-) or a plus (+) character. The minus character (-) indicates that years count in the negative direction when moving from the start date to the end date. The plus character (+) indicates that years count in the positive direction when moving from the start date to the end date. • *offset* — Specifies a number representing the first year of the era corresponding to the %Ey field descriptor. • *start_date* — Specifies the starting date of the era in yyyy/mm/dd format, where yyyy, mm, and dd are the year, month, and day, respectively, on the Gregorian calendar. Years prior to the year A.D. 1 are represented as negative numbers. For example, an era beginning March 5 in the year 100 B.C. would be represented as -100/03/05. • *end_date* — Specifies the ending date of the era in the same form used for the start_date variable or one of the two special values -* or +*. A -* value indicates that the ending date of the era extends backward to the beginning of time. A +* value indicates that the ending date of the era extends forward to the end of time. Therefore, the ending date can be chronologically before or after the starting date of the era. For example, the strings for the Christian eras A.D. and B.C. would be entered, respectively, in the following way: +:0:0000/01/01:+*:AD:%Ey %EC +:1:-0001/12/31:-*:BC:%Ey %EC • *name* — Specifies a string representing the name of the era that is substituted for the %EC field descriptor. • *format* — Specifies a strftime, strptime, and wcsftime format string to use when formatting the %EY field descriptor. This string can contain any strftime, strptime, and wcsftime format control characters (except %EY) and locale-dependent multibyte characters.

(continued on next page)

Table B–10 (Cont.) LC_TIME Category Keywords

Keyword	Description
	An era value consists of one string (enclosed in quotes) for each era. If more than one era is specified, each era string is separated by a semicolon (;).
era_d_fmt	Defines the string used to represent the date in alternate-era format corresponding to the %Ex field descriptor. The string can contain any combination of characters and field descriptors.
era_t_fmt	Defines the locale's alternative time format as represented by the %EX field descriptor for strftime, strptime, and wcsftime.
era_d_t_fmt	Defines the locale's alternative date-and-time format as represented by the %Ec field descriptor for strftime, strptime, and wcsftime.
alt_digits	Defines alternate strings for digits corresponding to the %O field descriptor.
	Recognized values consist of a group of strings separated by semicolons. The first string represents the alternate string for 0 (zero), the second string represents the alternate string for 1, and so on. You can specify a maximum of 100 alternate strings.

B.11.7.2 Field Descriptors

The LC_TIME locale definition source file uses field descriptors to represent elements of time and date formats. You can combine these field descriptors to create other field descriptors or to create time and date format strings. When used in format strings that contain field descriptors and other characters, field descriptors are replaced by their current values. All other characters are copied without change. Table B–11 lists the field descriptors used by commands and subroutines that query the LC_TIME category for time formatting.

Table B–11 LC_TIME Locale Field Descriptors

Field Descriptor	Meaning
%a	Represents the abbreviated weekday name (for example, Sun) defined by the abday statement.
%A	Represents the full weekday name (for example, Sunday) defined by the day statement.
%b	Represents the abbreviated month name (for example, Jan) defined by the abmon statement.
%B	Represents the full month name (for example, January) defined by the month statement.
%c	Represents the date-and-time format defined by the d_t_fmt statement.
%C	Represents the century as a decimal number (00 to 99).
%d	Represents the day of the month as a decimal number (01 to 31).
%D	Represents the date in %m/%d/%y format (for example, 01/31/91).
%e	Represents the day of the month as a decimal number (1 to 31).
	If the day of the month is not a 2-digit number, the leading digit is filled with a space character.

(continued on next page)

Table B–11 (Cont.) LC_TIME Locale Field Descriptors

Field Descriptor	Meaning
%Ec	Specifies the alternate date-and-time representation for the locale.
%EC	Specifies the name of the base year (period) in the locale's alternate representation.
%Ex	Specifies the alternate date representation for the locale.
%Ey	Specifies the offset from %EC (year only) in the locale's alternate representation.
%EY	Specifies the full alternate year representation.
%h	Represents the abbreviated month name (for example, Jan) defined by the `abmon` statement. This field descriptor is a synonym for the %b field descriptor.
%H	Represents the 24-hour clock hour as a decimal number (00 to 23).
%I	Represents the 12-hour clock hour as a decimal number (01 to 12).
%j	Represents the day of the year as a decimal number (001 to 366).
%m	Represents the month of the year as a decimal number (01 to 12).
%M	Represents the minutes of the hour as a decimal number (00 to 59).
%n	Specifies a new-line character.
%Od	Specifies the day of the month by using the locale's alternate numeric symbols.
%Oe	Specifies the day of the month by using the locale's alternate numeric symbols.
%OH	Specifies the hour (24-hour clock) by using the locale's alternate numeric symbols.
%OI	Specifies the hour (12-hour clock) by using the locale's alternate numeric symbols.
%Om	Specifies the month by using the locale's alternate numeric symbols.
%OM	Specifies the minutes by using the locale's alternate numeric symbols.
%OS	Specifies the seconds by using the locale's alternate numeric symbols.
%OU	Specifies the week number of the year (with Sunday as the first day of the week) by using the locale's alternate numeric symbols.
%Ow	Specifies the weekday as a number in the locale's alternate representation (Sunday = 0).
%OW	Specifies the week number of the year (with Monday as the first day of the week) by using the locale's alternate numeric symbols.
%Oy	Specifies the year (offset from %C) using the locale's alternate numeric symbols.
%p	Represents the a.m. or p.m. string defined by the `am_pm` statement.
%r	Represents the 12-hour clock time with a.m./p.m. notation as defined by the `t_fmt_ampm` statement.
%S	Represents the seconds of the minute as a decimal number (00 to 59).
%t	Specifies a tab character.
%T	Represents 24-hour clock time in the format %H:%M:%S (for example, 16:55:15).

(continued on next page)

Table B–11 (Cont.) LC_TIME Locale Field Descriptors

Field Descriptor	Meaning
%U	Represents the week of the year as a decimal number (00 to 53). Sunday, or its equivalent as defined by the day statement, is considered the first day of the week for calculating the value of this field descriptor.
%w	Represents the day of the week as a decimal number (0 to 6). Sunday, or its equivalent as defined by the day statement, is considered to be 0 (zero) for calculating the value of this field descriptor.
%W	Represents the week of the year as a decimal number (00 to 53). Monday, or its equivalent as defined by the day statement, is considered the first day of the week for calculating the value of this field descriptor.
%x	Represents the date format defined by the d_fmt statement.
%X	Represents the time format defined by the t_fmt statement.
%y	Represents the year of the century (00 to 99).
%Y	Represents the year as a decimal number (for example, 1989).
%%	Specifies a % (percent sign) character.

B.11.7.3 Sample Locale Definition

The following is a sample LC_TIME category specified in a locale definition source file:

```
LC_TIME
#
#Abbreviated weekday names (%a)
abday   "<S><u><n>";"<M><o><n>";"<T><u><e>";"<W><e><d>";\
        "<T><h><u>";"<F><r><i>";"<S><a><t>"

#Full weekday names (%A)
day     "<S><u><n><d><a><y>";"<M><o><n><d><a><y>";\
        "<T><u><e><s><d><a><y>";"<W><e><d><n><e><s><d><a><y>";\
        <T><h><u><r><s><d><a><y>";"<F><r><i><d><a><y>";\
        <S><a><t><u><r><d><a><y>"

#Abbreviated month names (%b)
abmon   "<J><a><n>";"<F><e><b>";"<M><a><r>";"<A><p><r>";\
        "<M><a><y>";"<J><u><n>";"<J><u><l>";"<A><u><g>";\
        <S><e><p>";"<O><c><t>";"<N><o><v>";"<D><e><c>"

#Full month names (%B)
mon     "<J><a><n><u><a><r><y>";"<F><e><b><r><u><a><r><y>";\
        "<M><a><r><c><h>";"<A><p><r><i><l>";"<M><a><y>";\
        <J><u><n><e>";"<J><u><l><y>";"<A><u><g><u><s><t>";\
        "<S><e><p><t><e><m><b><e><r>";"<O><c><t><o><b><e><r>";\
        <N><o><v><e><m><b><e><r>";"<D><e><c><e><m><b><e><r>"

#Date-and-time format (%c)
#Note that for improved readability, this section uses actual
#characters, rather than symbolic names, and is inconsistent with
#the other sections in this example.  This is bad form.
#In practice, symbolic names should be used.
d_t_fmt         "%a %b %d %H:%M:%S %Y"
#
#Date format (%x)
d_fmt           "%m/%d/%y"
#
#Time format (%X)
t_fmt           "%H:%M:%S"
```

```
#
#Equivalent of AM/PM (%p)
am_pm           "<A><M>";"<P><M>"
#
#12-hour time format (%r)
#Note that for improved readability, this section uses actual
#characters, rather than symbolic names, and is inconsistent with
#the other sections in this example.  This is bad form.
#In practice, symbolic names should be used.
t_fmt_ampm      "%I:%M:%S %p"
#
era             "+:0:0000/01/01:+*:AD:%Ey %EC";\
"+:1:-0001/12/31:-*:BC:%Ey %EC"

era_d_fmt     ""
alt_digits    "<0><t><h>";"<1><s><t>";"<2><n><d>";"<3><r><d>";\
              "<4><t><h>";"<5><t><h>";"<6><t><h>";"<7><t><h>";\
              "<8><t><h>";"<9><t><h>";"<1><0><t><h>"
#
END LC_TIME
```

B.12 Character Set Description (Charmap) File

This section describes the character set description file, or charmap file. The charmap file defines character symbols as character encodings and is the source file for a coded character set, or codeset.

B.12.1 Portable Character Set

All supported codesets have the Portable Character Set (PCS) as a proper subset. The PCS consists of the following character symbols (listed by their standardized symbolic names) and their hexadecimal encodings. See Table B–12.

Table B–12 Portable Character Set

Symbol Name	Hexadecimal Encoding
<NUL>	\x00
<alert>	\x07
<backspace>	\x08
<tab>	\x09
<newline>	\x0A
<vertical-tab>	\x0B
<form-feed>	\x0C
<carriage-return>	\x0D
<space>	\x20
<exclamation-mark>	\x21
<quotation-mark>	\x22
<number-sign>	\x23
<dollar-sign>	\x24
<percent>	\x25
<ampersand>	\x26

(continued on next page)

Table B–12 (Cont.) Portable Character Set

Symbol Name	Hexadecimal Encoding
<apostrophe>	\x27
<left-parenthesis>	\x28
<right-parenthesis>	\x29
<asterisk>	\x2A
<plus-sign>	\x2B
<comma>	\x2C
<hyphen>	\x2D
<period>	\x2E
<slash>	\x2F
<zero>	\x30
<one>	\x31
<two>	\x32
<three>	\x33
<four>	\x34
<five>	\x35
<six>	\x36
<seven>	\x37
<eight>	\x38
<nine>	\x39
<colon>	\x3A
<semi-colon>	\x3B
<less-than>	\x3C
<equal-sign>	\x3D
<greater-than>	\x3E
<question-mark>	\x3F
<commercial-at>	\x40
<A>	\x41
<B>	\x42
<C>	\x43
<D>	\x44
<E>	\x45
<F>	\x46
<G>	\x47
<H>	\x48
<I>	\x49
<J>	\x4A
<K>	\x4B
<L>	\x4C

(continued on next page)

Table B–12 (Cont.) Portable Character Set

Symbol Name	Hexadecimal Encoding
<M>	\x4D
<N>	\x4E
<O>	\x4F
<P>	\x50
<Q>	\x51
<R>	\x52
<S>	\x53
<T>	\x54
<U>	\x55
<V>	\x56
<W>	\x57
<X>	\x58
<Y>	\x59
<Z>	\x5A
<left-bracket>	\x5B
<backslash>	\x5C
<right-bracket>	\x5D
<circumflex>	\x5E
<underscore>	\x5F
<grave-accent>	\x60
<a>	\x61
<b>	\x62
<c>	\x63
<d>	\x64
<e>	\x65
<f>	\x66
<g>	\x67
<h>	\x68
<i>	\x69
<j>	\x6A
<k>	\x6B
<l>	\x6C
<m>	\x6D
<n>	\x6E
<o>	\x6F
<p>	\x70
<q>	\x71
<r>	\x72

(continued on next page)

Table B–12 (Cont.) Portable Character Set

Symbol Name	Hexadecimal Encoding
<s>	\x73
<t>	\x74
<u>	\x75
<v>	\x76
<w>	\x77
<x>	\x78
<y>	\x79
<z>	\x7A
<left-brace>	\x7B
<vertical-line>	\x7C
<right-brace>	\x7D
<tilde>	\x7E

B.12.2 Components of a Charmap File

A charmap file has the following components:

- An optional special symbolic name declarations section

  Each declaration in this section consists of a special symbolic name, followed by one or more space or tab characters, and a value. The following list describes the special symbolic names that you can include in the declarations section:

  `<code_set_name>`

  Specifies the name of the codeset for which the charmap file is defined. This value determines the value returned by the `nl_langinfo (CODESET)` subroutine. If <code_set_name> is not declared, the name for the Portable Character Set is used.

  `<mb_cur_max>`

  Specifies the maximum number of bytes in a character for the codeset. Valid values are 1 to 4. The default value is 1.

  `<mb_cur_min>`

  Specifies the minimum number of bytes in a character for the codeset. Since all supported codesets have the Portable Character Set as a proper subset, this value must be 1.

  `<escape_char>`

  Specifies the escape character that indicates encodings in hexadecimal or octal notation. The default value is a backslash (\).

  `<comment_char>`

  Specifies the character used to indicate a comment within a charmap file. The default value is the number sign (#).

- The CHARMAP section header

This header marks the beginning of the section that associates character symbols with encodings.

- Mapping statements for characters in the codeset

  Each statement specifies a symbolic name for a character and the associated encoding for that character. A mapping statement has the following format:

  <char_symbol> encoding

  A symbolic name begins with the left angle-bracket (<) character and ends with the right angle-bracket (>) character. For *char_symbol* (the name between < and >), you can use any characters from the Portable Character Set, except for control and space characters. You can use a > in *char_symbol*; if you do, precede all > characters except the last one with the escape character (as specified by the <escape_char> special symbolic name).

  An encoding is specified as one or more character constants, with the maximum number of character constants specified by the <mb_cur_max> special symbolic name. The encoding may be specified as decimal, octal, or hexadecimal constants with the following formats:

  - Decimal constant: \d*nn* or \d*nnn*, where *n* is any decimal digit
  - Octal constant: *nn* or *nnn*, where *n* is any octal digit
  - Hexadecimal constant: \x*nn*, where *n* is any hexadecimal digit

  The following are sample character symbol definitions:

```
<A>         \d65        #decimal constant
<B>         \x42        #hexadecimal constant
<j10101>    \x81\xA1    #multiple hexadecimal constants
```

  You can also define a range of symbolic names and corresponding encoded values, where the nonnumeric prefix for each symbolic name is common, and the numeric portion of the second symbolic name is equal to or greater than the numeric portion of the first symbolic name. In this format, a symbolic name value consists of zero or more nonnumeric characters followed by an integer of one or more decimal digits. This format defines a series of symbolic names. For example, the string <j0101>...<j0104> is interpreted as the symbolic names <j0101>, <j0102>, <j0103>, and <j0104>, in that order.

  In statements defining ranges of symbolic names, the specified encoded value is the value for the first symbolic name in the range. Subsequent symbolic names have encoded values in increasing order. Consider the following sample statement:

```
<j0101>...<j0104>        \d129\d254
```

  This sample statement is interpreted as follows:

```
<j0101> \d129\d254
<j0102> \d129\d255
<j0103> \d130\d0
<j0104> \d130\d1
```

  You cannot assign multiple encodings to one symbolic name, but you can create multiple names for one encoded value because some characters have several common names. For example, the . character is called a period in some parts of the world, and a full stop in others. You can specify both names in the charmap. For example:

```
<period>        \x2e
<full-stop>     \x2e
```

Any comments must begin with the character specified by the <comment_char> special symbolic name. When an entire line is a comment, you must specify the <comment_char> in the first column of the line.

- The END CHARMAP section trailer

  This trailer indicates the end of character map statements.

The following is a portion of a sample charmap file:

```
CHARMAP
<code_set_name>         "ISO8859-1"
<mb_cur_max>            1
<mb_cur_min>            1
<escape_char>           \
<comment_char>          #

<NUL>                   \x00
<SOH>                   \x01
<STX>                   \x02
<ETX>                   \x03
<EOT>                   \x04
<ENQ>                   \x05
<ACK>                   \x06
<alert>                 \x07
<backspace>             \x08
<tab>                   \x09
<newline>               \x0a
<vertical-tab>          \x0b
<form-feed>             \x0c
<carriage-return>       \x0d
END CHARMAP
```

C

SCSI as a VMScluster Storage Interconnect—OpenVMS Alpha Version 6.2 Feature

One of the benefits of VMScluster systems is that multiple computers can simultaneously access storage devices connected to a VMScluster storage interconnect. Together, these systems provide high performance and highly available access to storage.

This appendix describes how VMScluster systems support the Small Computer Systems Interface (SCSI) as a storage interconnect. Multiple Alpha computers, also referred to as hosts or nodes, can simultaneously access SCSI disks over a SCSI interconnect. A SCSI interconnect, also called a SCSI bus, is an industry-standard interconnect that supports one or more computers, peripheral devices, and interconnecting components.

The discussions in this chapter assume that you already understand the concept of sharing storage resources in a VMScluster environment. VMScluster concepts and configuration requirements are described in the following VMScluster documentation:

- *Guidelines for VMScluster Configurations*
- *VMScluster Systems for OpenVMS*
- OpenVMS Cluster Software *Software Product Description* (SPD 29.78.*xx*)

This appendix includes two primary parts:

- Section C.1 through Section C.6.6 describe the fundamental procedures and concepts that you would need to plan and implement a SCSI VMScluster system.
- Section C.7 and its subsections provide additional technical detail and concepts; these sections can be seen as containing supplementary information about SCSI VMScluster systems that would typically be located in an appendix.

C.1 Conventions Used in This Appendix

Certain conventions are used throughout this appendix to identify the ANSI Standard and for elements in figures.

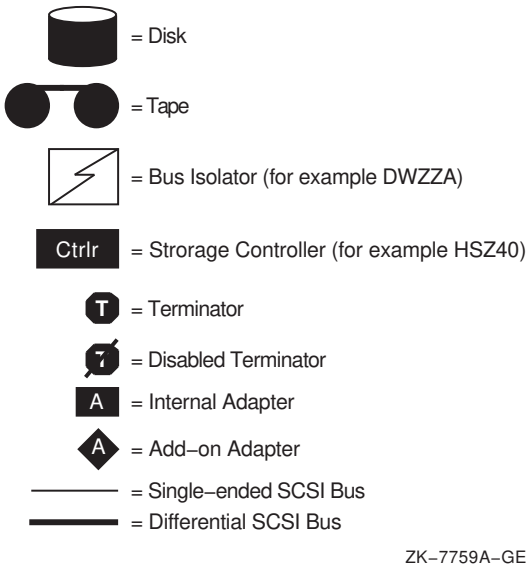
C.1.1 SCSI ANSI Standard

VMScluster systems configured with the SCSI interconnect must use standard SCSI–2 components. The SCSI–2 components supported must be compliant with the architecture defined in the *American National Standards Institute (ANSI) Standard SCSI–2*. This standard defines extensions to the SCSI–1 standard. For ease of discussion, this appendix uses the term SCSI or SCSI–2 to refer to the SCSI–2 implementation as specified in the ANSI Standard SCSI–2 document X3T9.2, Rev. 10L.

C.1.2 Symbols Used in Figures

Figure C–1 is a key to the symbols used in figures throughout this appendix.

Figure C–1 Conventions: Key to Symbols Used in Figures



C.2 Accessing SCSI Storage

In VMScluster configurations, multiple VAX and Alpha hosts can directly access SCSI devices in any of the following ways:

- CI interconnect with HSJ or HSC controllers
- Digital Storage Systems Interconnect (DSSI) with HSD controller
- SCSI adapters directly connected to VAX or Alpha systems

You can also access SCSI devices indirectly using the OpenVMS MSCP server.

The following sections describe single-host and multiple-host access to SCSI storage devices.

C.2.1 Single-Host SCSI Access in VMScluster Systems

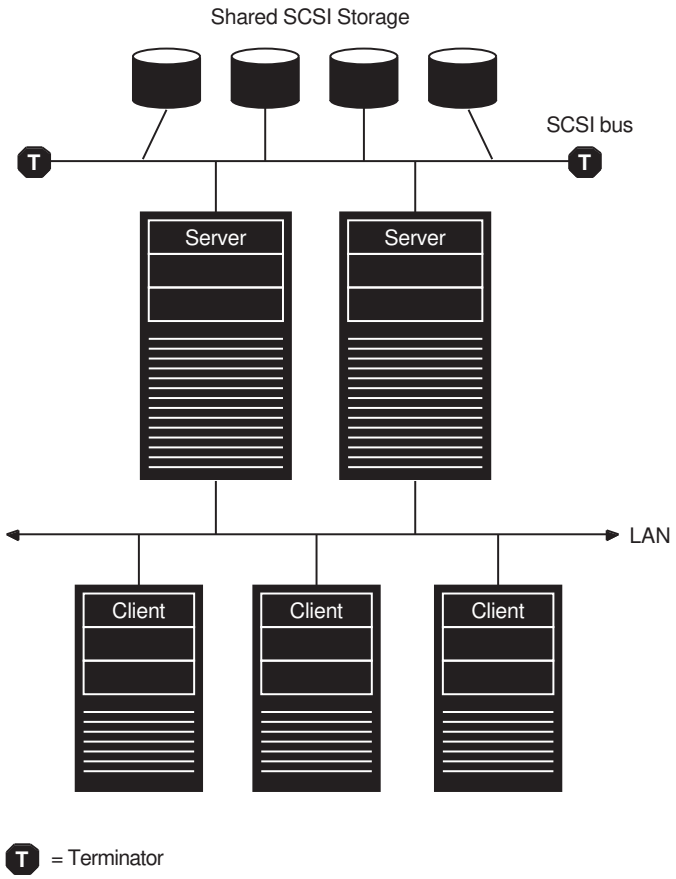
Prior to OpenVMS Version 6.2, VMScluster systems provided support for SCSI storage devices connected to a single host using an embedded SCSI adapter, an optional external SCSI adapter, or a special-purpose RAID (redundant array of independent disks) controller. Only one host could be connected to a SCSI bus.

C.2.2 Multiple-Host SCSI Access in VMScluster Systems

Beginning with OpenVMS Alpha Version 6.2, multiple Alpha hosts in a VMScluster system can be connected to a single SCSI bus to share access to SCSI storage devices directly. This capability allows you to build highly available servers using shared access to SCSI storage.

Figure C–2 shows a VMScluster configuration that uses a SCSI interconnect for shared access to SCSI devices. Note that another interconnect (for example, a local area network [LAN]) is required for host-to-host VMScluster (System Communications Architecture [SCA]) communications.

Figure C–2 Highly Available Servers for Shared SCSI Access



You can build a three-node VMScluster system using the shared SCSI bus as the storage interconnect, or you can include shared SCSI buses within a larger VMScluster configuration. A quorum disk can be used on the SCSI bus to improve the availability of two- or three-node configurations. Host-based RAID (including host-based shadowing) and the MSCP server are supported for shared SCSI storage devices.

C.3 Configuration Requirements and Hardware Support

This section lists the configuration requirements and supported hardware for SCSI VMScluster systems for OpenVMS Version 7.0.

C.3.1 Configuration Requirements

Table C–1 shows the requirements and capabilities of the basic software and hardware components you can configure in a SCSI VMScluster system.

Table C–1 Requirements for SCSI VMScluster Configurations

Requirement	Description
Software	All Alpha hosts sharing access to storage on a SCSI interconnect must be running: • OpenVMS Alpha Version 6.2 or later • VMScluster Software for OpenVMS Alpha Version 6.2 or later
Hardware	Table C–2 lists the supported hardware components for SCSI VMScluster systems. See also Section C.7.7 for information about other hardware devices that might be used in a SCSI VMScluster configuration.
SCSI tape, floppies, and CD–ROM drives	You cannot configure SCSI tape drives, floppy drives, or CD–ROM drives on multiple-host SCSI interconnects. If your configuration requires SCSI tape, floppy, or CD–ROM drives, configure them on single-host SCSI interconnects. Note that SCSI tape, floppy, or CD–ROM drives may be MSCP or TMSCP served to other hosts in the VMScluster configuration.
Maximum hosts on a SCSI bus	You can connect up to three hosts on a multiple-host SCSI bus. You can configure any mix of the hosts listed in Table C–2 on the same shared SCSI interconnect.
Maximum SCSI buses per host	You can connect each host to a maximum of three multiple-host SCSI buses. The number of nonshared (single-host) SCSI buses that can be configured is limited only by the number of available slots on the host bus.
Host-to-host communication	All members of the cluster must be connected by an interconnect that can be used for host-to-host (SCA) communication; for example, DSSI, CI, Ethernet, or FDDI.
Host-based RAID (including host-based shadowing)	Supported in SCSI VMScluster configurations.
SCSI device naming	The name of each SCSI device must be unique throughout the VMScluster system. When configuring devices on systems that include a multiple-host SCSI bus, adhere to the following requirements: • A host can have, at most, one controller attached to a particular SCSI interconnect. • All host controllers attached to a given SCSI interconnect must have the same OpenVMS device name (for example, PKA0). • Each system attached to a SCSI interconnect must have the same nonzero allocation class. Each disk device name (for example, DKB100), whether or not on a shared SCSI bus, must be unique within an allocation class. Refer to Section C.6.2 for more information.

C.3.2 Hardware Support

Table C–2 shows the supported hardware components for SCSI VMScluster systems; it also lists the minimum required revision for these hardware components. That is, for any component, you must use either the version listed in Table C–2 or a subsequent version.

The SCSI interconnect configuration and all devices on the SCSI interconnect must meet the requirements defined in the *ANSI Standard SCSI–2* document and the requirements described in this appendix.)

See also Section C.7.7 for information about other hardware devices that might be used in a SCSI VMScluster configuration.

Table C–2 Supported Hardware for SCSI VMScluster Systems

Component	Supported Item	Minimum Version or H/W Revision	How to Find Your Version
Hosts	AlphaServer 400	See footnote[1]	Console SHOW VERSION command
	AlphaServer 1000	See footnote[1]	
	AlphaServer 2000	See footnote[1]	
	AlphaServer 2100	See footnote[1]	
	AlphaStation 200	See footnote[1]	
	AlphaStation 250	See footnote[1]	
	AlphaStation 400	See footnote[1]	
	AlphaStation 600	See footnote[1]	
Disks[3]	RZ26	n/a	Console SHOW DEVICE command
	RZ26L	442D	
	RZ26N	n/a	
	RZ28	442D	
	RZ28B	0006	
	RZ28M	n/a	
	RZ29B	0006	
Controller	HSZ40–B	2.5	Console SHOW DEVICE command
Bus Isolators	DWZZA–AA	E01	Examine product sticker
	DWZZA–VA	F01	

[1]The minimum revision of this component for SCSI VMScluster configurations is the version included in the Version 3.2 Firmware Kit, on the May, 1995 CD–ROM. The revision number is listed in the *Firmware Release Notes Overview* that accompanies that Firmware Kit.

[3]Both wide and narrow variants have been qualified. However, the KZPAA is a narrow-mode adapter and does not enable wide-mode operation.

(continued on next page)

Table C–2 (Cont.) Supported Hardware for SCSI VMScluster Systems

Component	Supported Item	Minimum Version or H/W Revision	How to Find Your Version
Adapters[2]	Integral system adapter	N/A	
	KZPAA (PCI to SCSI)		

[2]You can configure other types of SCSI adapters in a system for single-host access to local storage.

C.4 SCSI Interconnect Concepts

The SCSI standard defines a set of rules governing the interactions between initiators (typically, host systems) and SCSI targets (typically, peripheral devices). This standard allows the host to communicate with SCSI devices (such as disk drives, tape drives, printers, and optical media devices) without having to manage the device-specific characteristics.

The following sections describe the SCSI standard and the default modes of operation. The discussions also describe some optional mechanisms you can implement to enhance the default SCSI capabilities in areas such as capacity, performance, availability, and distance.

C.4.1 Number of Devices

The SCSI bus is an I/O interconnect that can support up to eight devices. The devices can include host adapters, peripheral controllers, and discrete peripheral devices such as disk or tape drives. The devices are addressed by a unique ID number from 0 through 7. You assign the device IDs by entering console commands, or by setting jumpers or switches.

To increase the number of devices on the SCSI interconnect, some devices implement a second level of device addressing using logical unit numbers (LUNs). For each device ID, up to eight LUNs (0–7) can be used to address a single SCSI device as multiple units. The maximum number of LUNs per device ID is eight.

Note

When connecting devices to a SCSI interconnect, each device on the interconnect must have a unique device ID. You may need to change a device's default device ID to make it unique. For information about setting a device's ID, refer to the owner's guide for the device.

C.4.2 Performance

The default mode of operation for all SCSI devices is 8-bit asynchronous mode. This mode, sometimes referred to as narrow mode, transfers 8 bits of data from one device to another. Each data transfer is acknowledged by the device receiving the data. Because the performance of the default mode is limited, the SCSI standard defines optional mechanisms to enhance performance. The following list describes two optional methods for achieving higher performance:

- Increase the amount of data that is transferred in parallel on the interconnect. The 16-bit and 32-bit wide options allow a doubling or quadrupling of the data rate, respectively. Because the 32-bit option is seldom implemented, this appendix discusses only 16-bit operation and refers to it using the term **wide**.

- Use synchronous data transfer. In synchronous mode, multiple data transfers can occur in succession, followed by an acknowledgment from the device receiving the data. The standard defines a Slow Mode (also called Standard Mode) and a Fast Mode for synchronous data transfers:
  - In Standard Mode, the interconnect achieves up to 5 million transfers per second.
  - In Fast Mode, the interconnect achieves up to 10 million transfers per second.

Because all communications on a SCSI interconnect occur between two devices at a time, each pair of devices must negotiate to determine which of the optional features they will use. Most, if not all, SCSI devices implement one or more of these options.

Table C–3 shows data rates when using 8- and 16-bit transfers with Standard and Fast synchronous modes.

Table C–3 Maximum Data Transfer Rates in Megabytes per Second

Mode	Narrow (8-Bit)	Wide (16-Bit)
Standard	5	10
Fast	10	20

C.4.3 Distance

The maximum length of the SCSI interconnect is determined by the signaling method used in the configuration and, in some cases, by the data transfer rate. There are two types of electrical signaling for SCSI interconnects:

- Single-ended signaling

  The single-ended method is the most common and the least expensive. It can operate in either Standard or Fast Mode. The mode used determines the length of the interconnect, as follows:
  - When standard transfers are in use, the interconnect can be up to 6 meters in length [1]
  - When fast transfers are in use, the interconnect can be up to 3 meters in length
- Differential signaling

  This method provides higher signal integrity, thereby allowing a SCSI bus to span distances of up to 25 meters. Differential signaling allows both standard and fast data transfers regardless of the length of the SCSI bus.

When considering cable distance issues, be sure to include both internal and external cabling in your calculations. Table C–5 lists the internal cable lengths for various configurations.

[1] See the note following Table C–4 for information about the maximum length recommended by Digital.

Table C–4 summarizes how the type of signaling method affects SCSI interconnect distances.

Table C–4 Maximum SCSI Interconnect Distances

Signaling Technique	Rate of Data Transfer	Maximum Cable Length
Single ended	Standard	6 meters†
Single ended	Fast	3 meters
Differential	Standard or Fast	25 meters

†The SCSI standard specifies a maximum length of 6 meters for this type of interconnect. However, Digital recommends that, where possible, you limit the cable length to 4 meters to ensure the highest level of data integrity.

A **DWZZA converter** is a single-ended to differential converter that you can use to connect single-ended and differential SCSI interconnect segments. The differential segments are useful for the following:

- Overcoming the distance limitations of the single-ended interconnect
- Allowing communication between single-ended and differential devices

Because the DWZZA is strictly a signal converter, you do not need to assign a SCSI device ID to it. You can configure a maximum of two DWZZA converters in the path between any two hosts.

C.4.4 Cabling and Termination

Each single-ended and differential SCSI interconnect must have two terminators, one at each end. The specified maximum interconnect lengths are measured from terminator to terminator.

The interconnect terminators are powered from the SCSI interconnect line called TERMPWR. Each Digital host adapter and enclosure supplies the TERMPWR interconnect line, so that as long as one host or enclosure is powered on, the interconnect remains terminated.

Devices attach to the interconnect by short cables (or etch), called stubs. Stubs must be short in order to maintain the signal integrity of the interconnect. The maximum stub lengths allowed are determined by the type of signaling used by the interconnect, as follows:

- For single-ended interconnects, the maximum stub length is .1 meters
- For differential interconnects, the maximum stub length is .2 meters

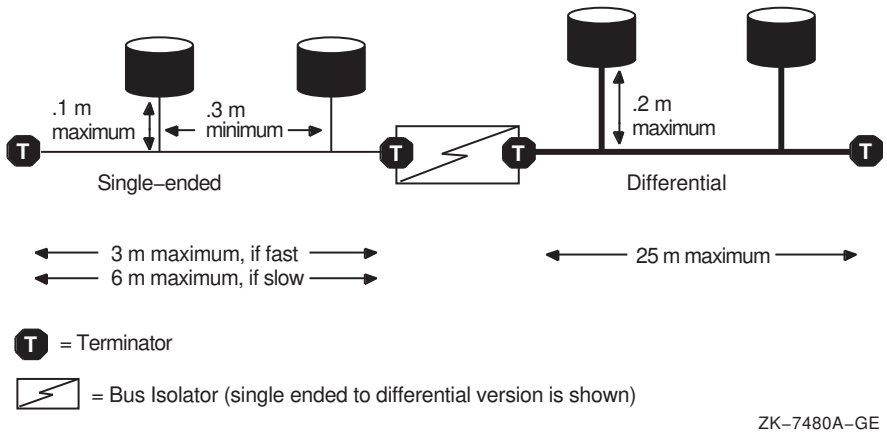
Additionally, the minimum distance between stubs on a single-ended interconnect is .3 meters. Refer to Figure C–3 for an example of this configuration.

Note

Terminate single-ended and differential buses individually, even when using DWZZA converters.

When you are extending the SCSI bus beyond an existing terminator, it is necessary to disable or remove that terminator.

Figure C–3 Maximum Stub Lengths



When using the host system's internal SCSI adapter, you must configure the system at the end of the single-ended SCSI segment. This is because the internal SCSI cable lengths exceed the allowable SCSI stub length. However, host systems are not required to be configured at the ends of the bus segment when an add-on SCSI adapter is used.

See Table C–5 for information about internal SCSI cable lengths.

C.5 SCSI VMScluster Hardware Configurations

The hardware configuration that you choose depends on a combination of factors:

- Your computing needs—for example, continuous availability, or the ability to disconnect or remove a system from your SCSI VMScluster system
- Your environment—for example, the physical attributes of your computing facility
- Your resources—for example, your capital equipment or the available PCI slots

You can connect up to three hosts on a shared SCSI interconnect. Each host can be connected to one, two, or three shared SCSI interconnects. The number of nonshared SCSI buses that can be configured is limited only by the number of available slots on the host bus.

The following sections provide guidelines for building SCSI configurations and describe potential configurations that might be suitable for various sites.

C.5.1 Systems Using Add-On SCSI Adapters

Shared SCSI bus configurations may use optional add-on KZPAA adapters. These adapters are generally easier to configure than internal adapters because they do not consume any SCSI cable length. Additionally, when you configure systems using KZPAA adapters for the shared SCSI bus, the internal adapter is available for connecting devices that cannot be shared (for example, SCSI tape, floppy, and CD–ROM drives).

When using KZPAA adapters, storage is configured using BA350, BA353, or HSZ40 StorageWorks enclosures. These enclosures are suitable for all data disks, and for shared VMScluster system and quorum disks. By using StorageWorks enclosures, it is possible to shut down individual systems without losing access to the disks.

The following sections describe some SCSI VMScluster configurations that take advantage of add-on adapters.

C.5.1.1 Building a Basic System Using Add-On SCSI Adapters

Figure C–4 shows a logical representation of a basic configuration using SCSI adapters and a StorageWorks enclosure. This configuration has the advantage of being relatively simple, while still allowing the use of tapes, floppies, CD–ROMs, and disks with nonshared files (for example, page files and swap files) on internal buses. Figure C–5 shows this type of configuration using AlphaServer 1000 systems and a BA350 enclosure.

The BA350 enclosure uses 0.9 meters of SCSI cabling, and this configuration typically uses two 1-meter SCSI cables. (A BA353 enclosure also uses 0.9 meters, with the same total cable length.) The resulting total cable length of 2.9 meters allows Fast SCSI Mode operation.

Although the shared BA350 storage enclosure is theoretically a single point of failure, this basic system is a very reliable SCSI VMScluster configuration. When the quorum disk is located in the BA350, you can shut down either of the AlphaStation systems independently while retaining access to the VMScluster system. However, you cannot physically remove the AlphaStation system, because that would leave an unterminated SCSI bus.

If you need the ability to remove a system while your VMScluster system remains operational, build your system using DWZZA converters, as described in Section C.5.1.2. If you need continuous access to data if a SCSI interconnect fails, you should do both of the following:

- Add a redundant SCSI interconnect with another BA350 shelf.
- Shadow the data.

In Figure C–4 and the other logical configuration diagrams in this appendix, the required network interconnect is not shown. (See Figure C–1 for the key to the symbols used in the following figures.)

Figure C–4 Conceptual View: Basic SCSI System

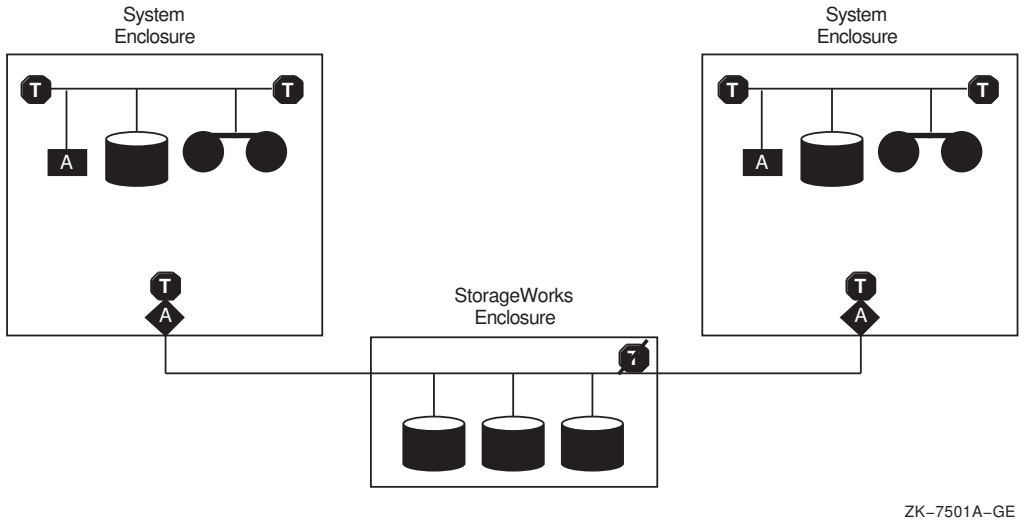
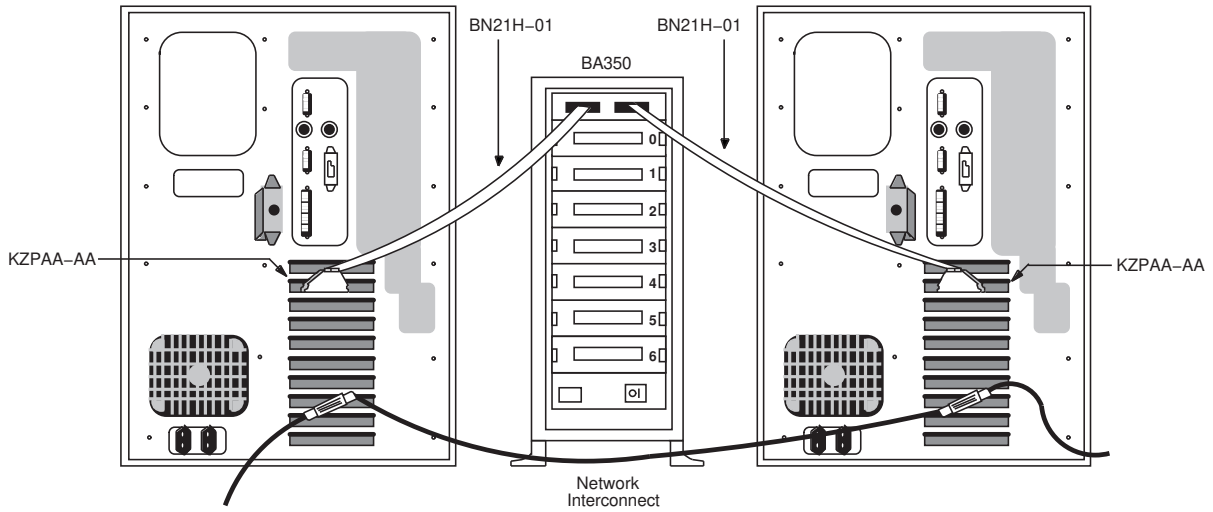


Figure C–5 Sample Configuration: Basic SCSI System Using AlphaServer 1000, KZPAA Adapter, and BA350 Enclosure



C.5.1.2 Building a System That Allows a Server to Be Removed (Using DWZZA Converters)

The capability of removing an individual system from your SCSI VMScluster configuration (for maintenance or repair) while the other systems in the cluster remain active gives you an especially high level of availability. To have this capability, use a configuration that includes a DWZZA converter (a SCSI bus isolator). DWZZA converters provide additional SCSI bus length capabilities, because the DWZZA allows you to connect a single-ended device to a bus that uses differential signaling. As described in Section C.4.3, SCSI bus configurations that use differential signaling may span distances up to 25 meters, whereas single-ended configurations can span only 3 meters when Fast Mode data transfer is used.

DWZZA converters are available as standalone, desktop components or as StorageWorks compatible building blocks. DWZZA converters can be used with the internal SCSI adapter or the optional KZPAA adapters.

Figure C–6 shows a logical view of a configuration that uses internal SCSI adapters and a pair of bus isolators, and Figure C–7 shows a physical view of the same configuration using two AlphaServer 1000 systems. In configurations such as those shown in Figure C–6 and Figure C–7, you can also remove either of the AlphaServer enclosures, because the SCSI bus remains terminated.

In each of these figures, note that a single-ended SCSI bus is used to connect a DWZZA to the AlphaServer systems, and another single-ended bus is used to connect the second DWZZA to the disks. The two DWZZAs are connected to each other by a differential bus. The differential signaling is necessary because the cabling between the DWZZA and the AlphaServer systems consumes virtually all of the 3 meters of single-ended cabling that is allowed for Fast Mode data transfer.

(See Figure C–1 for the key to the symbols used in the following figures.)

Figure C–6 Conceptual View: SCSI System with Bus Isolator (DWZZA Converter)

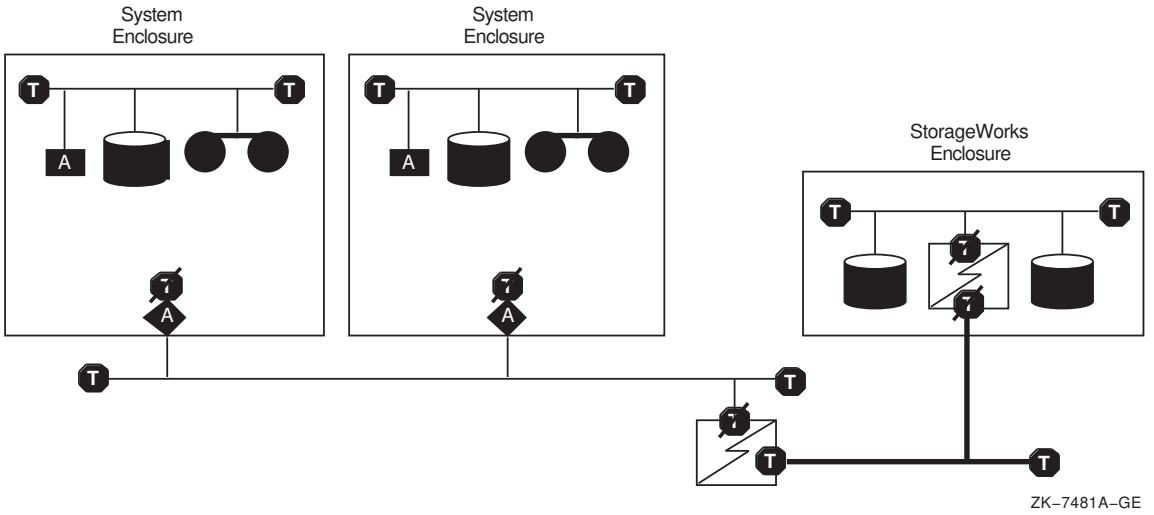
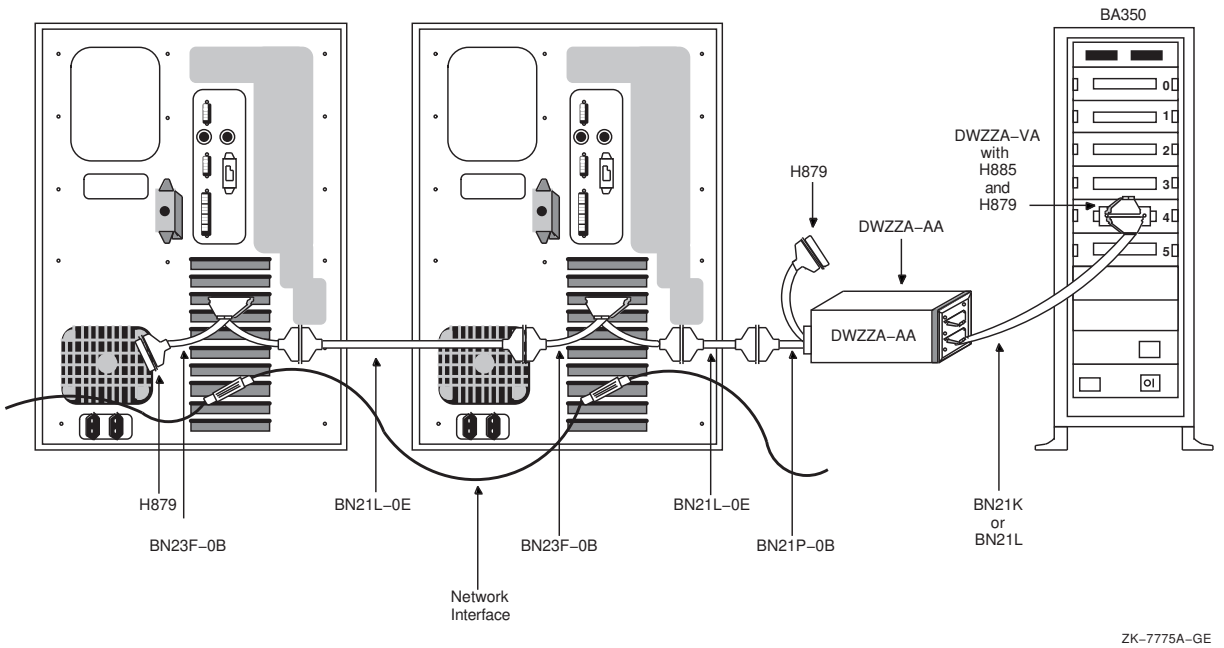


Figure C–7 Sample Configuration: SCSI System with DWZZA Converter, AlphaServer 1000 Systems, and BA350 Enclosure



C.5.1.3 Building a System That Allows Additional Features and Performance Using an HSZ40 Controller

The HSZ40 is a high-performance differential SCSI controller that can be connected to a differential SCSI bus, and supports up to 42 SCSI devices. An HSZ40 may be configured on a shared SCSI bus that includes DWZZA single-ended to differential converters. Disk devices configured on HSZ40 controllers can be combined into RAID sets to further enhance performance and provide high availability.

Figure C–8 shows a logical view of a configuration that uses differential SCSI controllers, and Figure C–9 shows a physical example of the same configuration using an HSZ40 in an SW300 enclosure. Note that the DWZZA is not needed in the StorageWorks enclosure in this type of configuration, because the HSZ40 is compatible with differential signaling.

(See Figure C–1 for the key to the symbols used in the following figures.)

Figure C–8 Conceptual View: System Using Differential Controllers

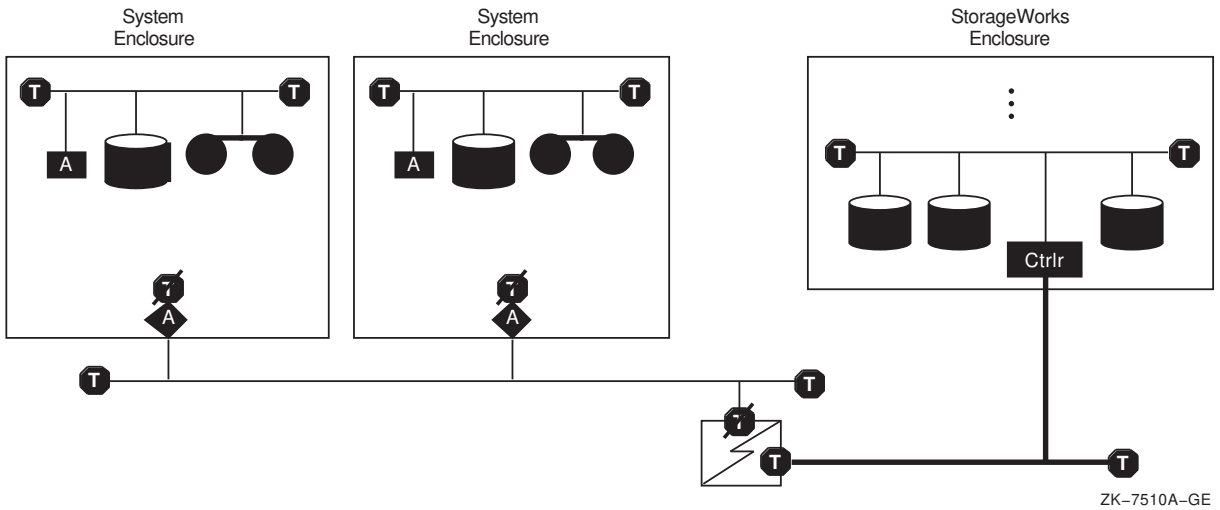
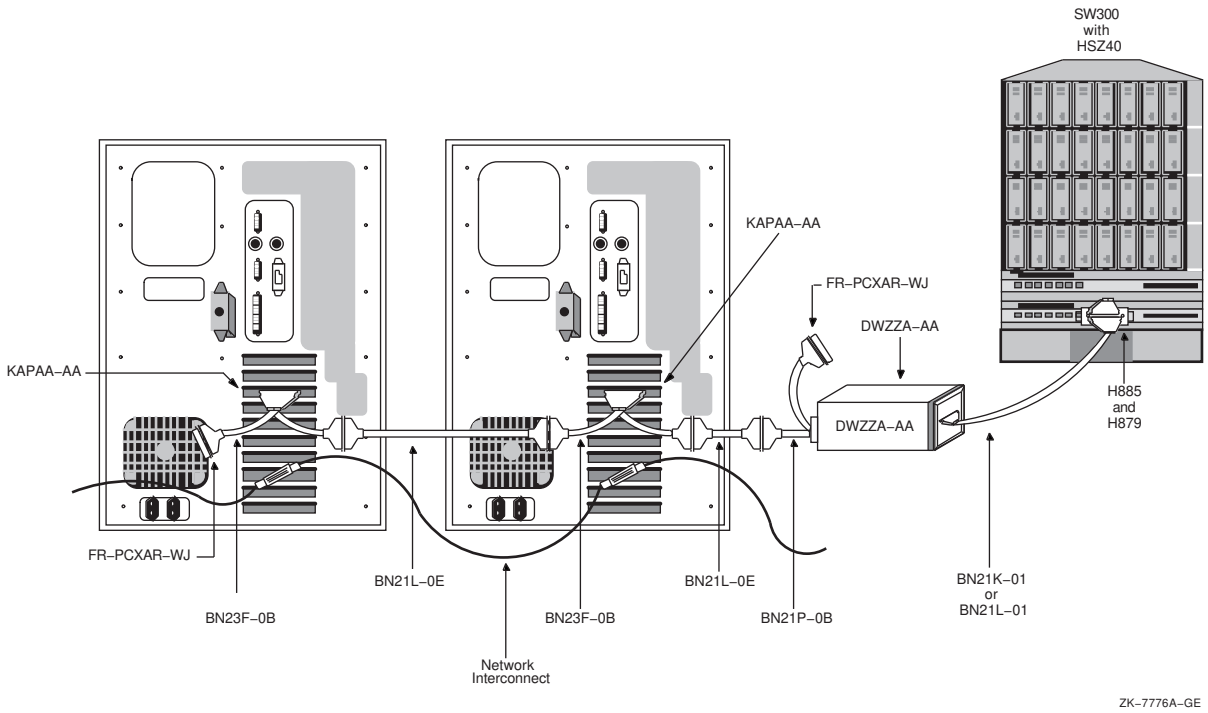


Figure C–9 Sample Configuration: System Using HSZ40 Controller in an SW300 Enclosure



C.5.1.4 Building a System with More Enclosures or Greater Separation

If you need additional enclosures, or if the needs of your site require a greater physical separation between systems, you can use a configuration in which DWZZAs are placed between systems with single-ended signaling and a differential-cabled SCSI bus.

Figure C–10 shows a logical view of a configuration that uses additional DWZZAs to increase the potential physical separation (or to allow for additional enclosures), and Figure C–11 shows a sample representation of this configuration.

Figure C–10 Conceptual View: Using DWZZAs to Allow for Increased Separation or More Enclosures

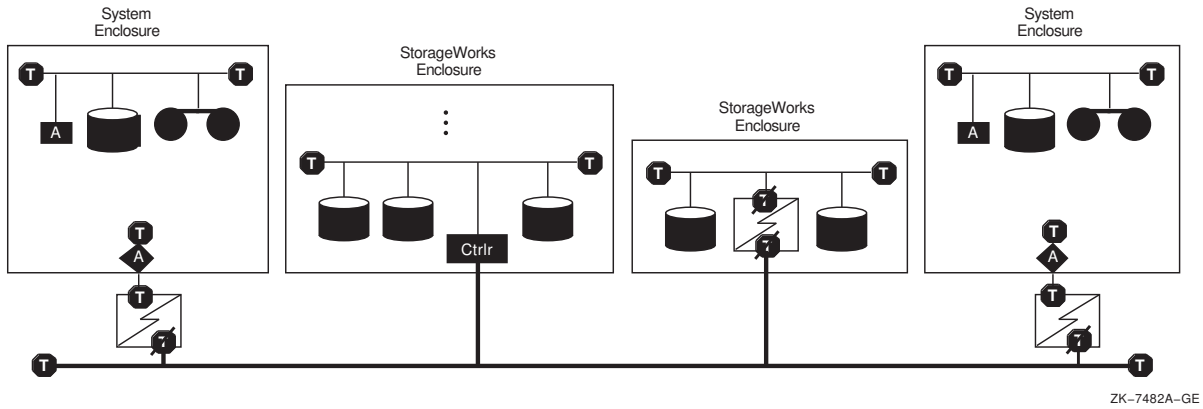
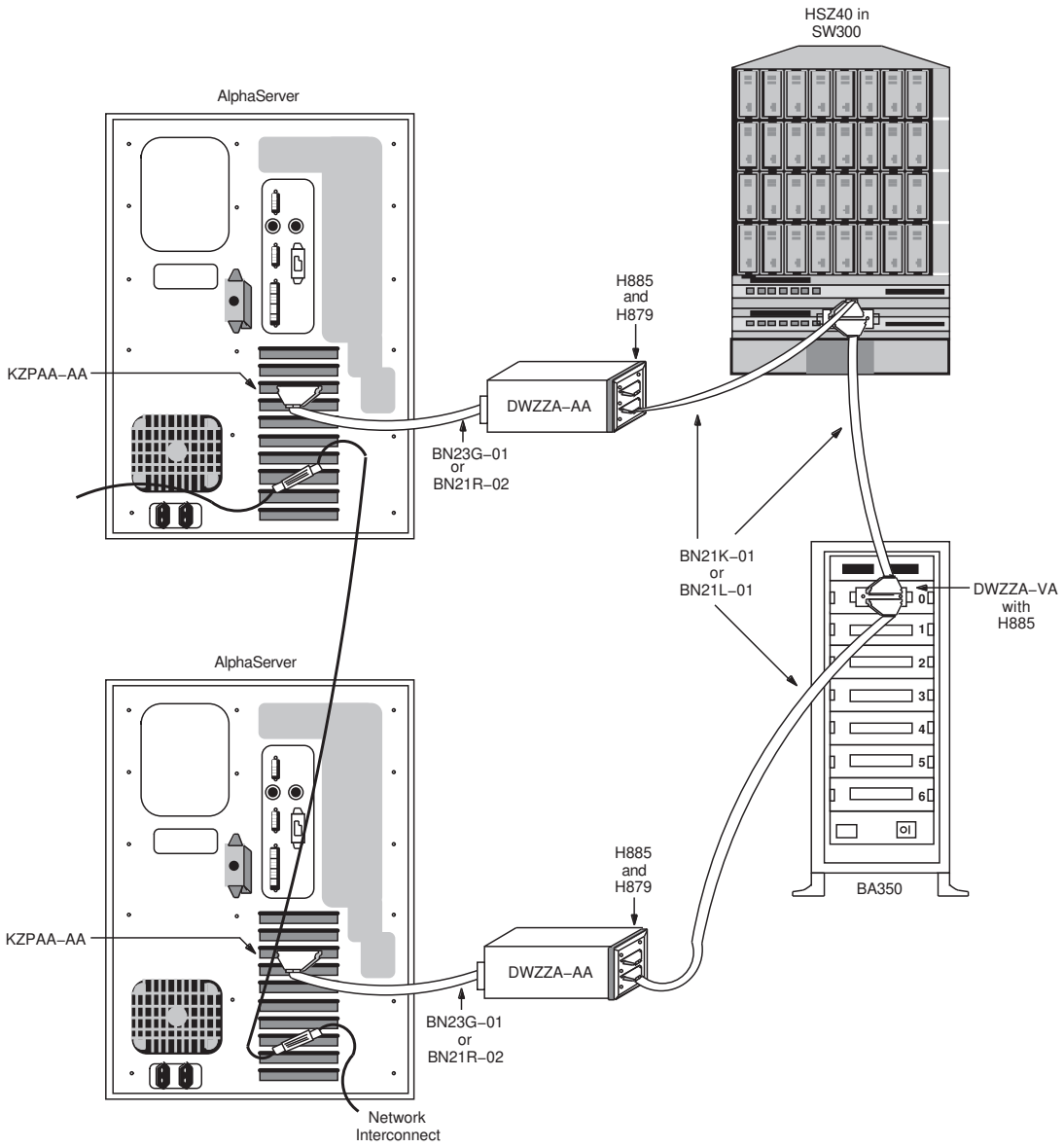
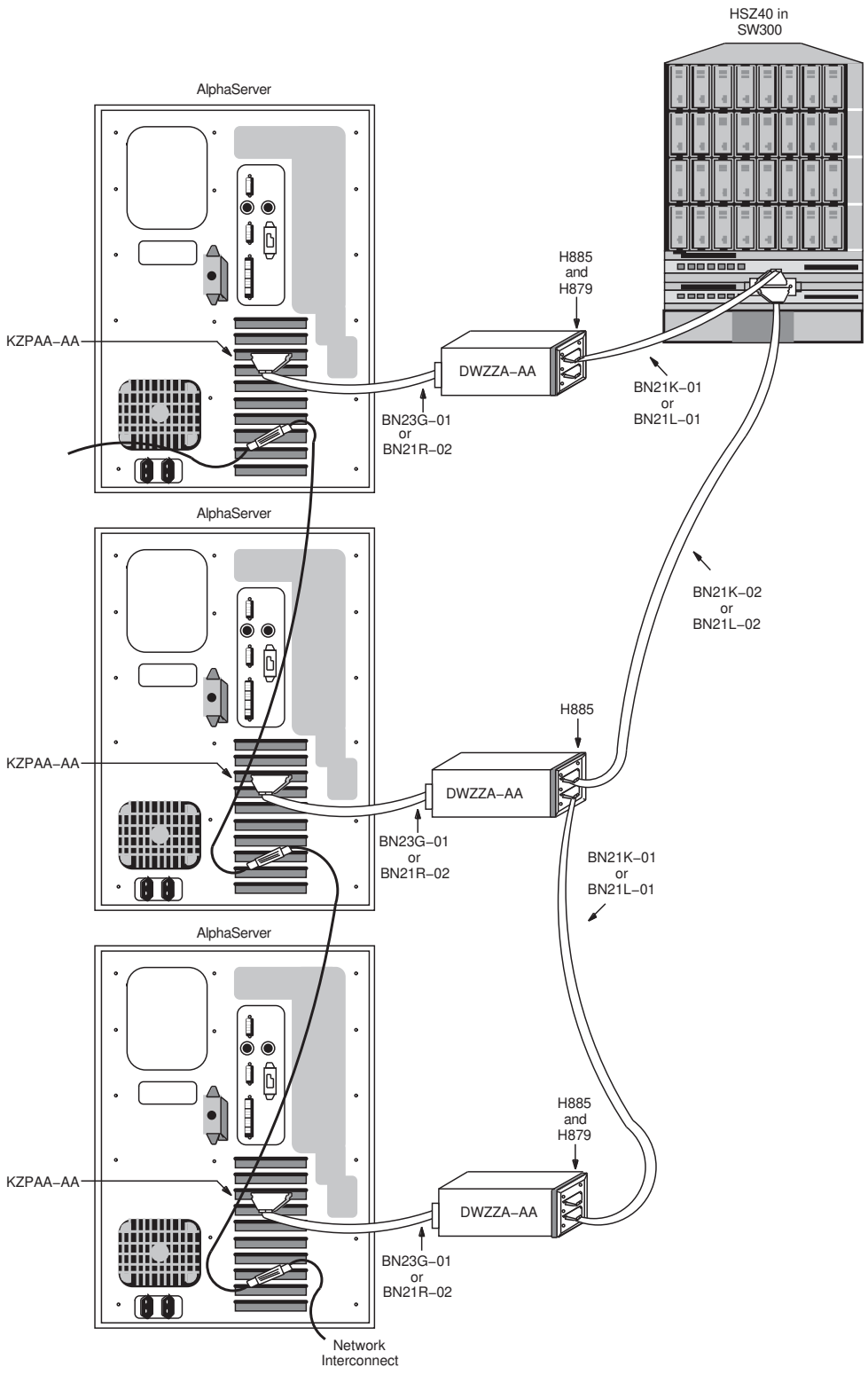


Figure C–11 Sample Configuration: Using DWZZAs to Allow for Increased Separation or More Enclosures



In OpenVMS Version 7.0, you can connect up to three hosts on a multiple-host SCSI bus. Figure C–12 shows how a three-host SCSI VMScluster system might be configured.

Figure C–12 Sample Configuration: Three Hosts on a SCSI Bus



C.5.2 Building a System Using Internal SCSI Adapters

You can build a multiple-host SCSI VMScluster configuration with two systems using internal adapters that are joined by a single SCSI cable. This type of configuration is relatively inexpensive, and it provides some of the benefits of multiple-host SCSI VMScluster systems that use external adapters (for example, fully-shared disks and twice the serving performance of a single system). This system configuration can also be expanded to provide improved performance, availability, and scaling.

However, a multiple-host SCSI VMScluster system that uses only internal SCSI adapters has the following limitations:

- On most systems, the internal cabling lengths exceeds 3 meters and therefore precludes the use of Fast Mode data transfer (see Table C–5).
- You cannot remove either of the AlphaServer systems for maintenance or repair while the remaining cluster systems stay active (because the bus would be unterminated).
- If these are the only members of the cluster, then quorum is lost when one (or either, depending on how the votes are allocated) of the enclosures goes down.
- You cannot use tape or CD–ROM drives.

Some of the limitations associated with the internal adapter can be removed by using DWZZAs, additional SCSI adapters, and additional storage enclosures.

Figure C–13 shows a conceptual view of a SCSI system using internal adapters, and Figure C–14 shows a sample configuration of such a system. (See Figure C–1 for the key to the symbols used in these figures.)

Figure C–13 Conceptual View: SCSI VMScluster System Using Internal Adapters

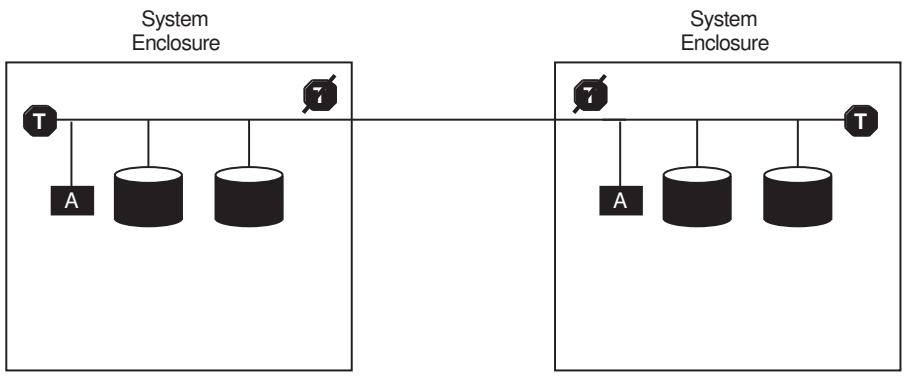


Figure C–14 Sample Configuration: SCSI VMScluster System with AlphaStation 200 Systems Using Internal Adapters

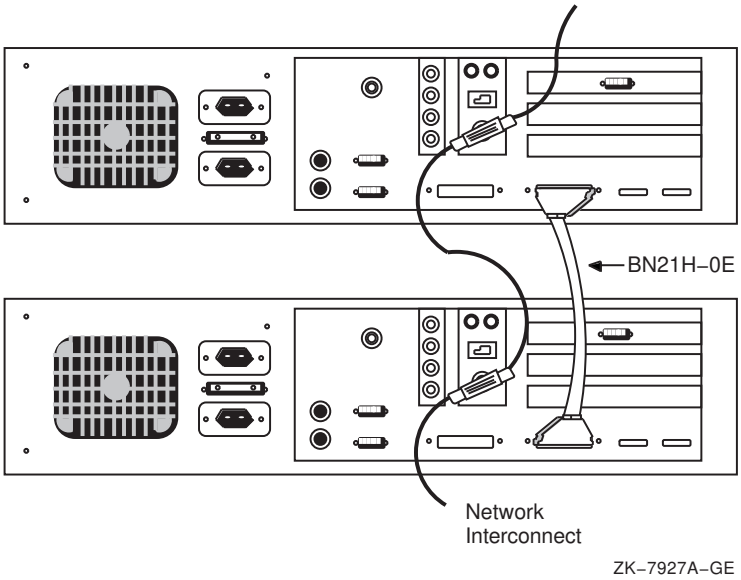


Table C–5 Internal SCSI Cable Lengths

System Type	Internal Cable Length
AlphaServer 1000 rackmount	1.6 meters
AlphaServer 1000 pedestal with an internal StorageWorks shelf in a dual-bus configuration[1]	2.0 meters
AlphaServer 2000 pedestal with the internal StorageWorks shelf that is not connected	1.7 meters
AlphaServer 2100 rackmount	2.0 meters
AlphaServer 2100 pedestal with the internal StorageWorks shelf that is not connected	1.6 meters
AlphaStation 200	1.2 meters
AlphaStation 400	1.4 meters

[1]See your hardware manual for an explanation of single-bus and dual-bus configurations, and how to switch from one to the other.

C.6 Installation

This section describes the steps required to set up and install the hardware in a SCSI VMScluster system. The assumption in this section is that a new VMScluster system, based on a shared SCSI bus, is being created. If, on the other hand, you are adding a shared SCSI bus to an existing VMScluster configuration, then you should integrate the procedures in this section with those described in *VMScluster Systems for OpenVMS* to formulate your overall installation plan.

Table C–6 lists the steps required to set up and install the hardware in a SCSI VMScluster system.

Table C–6 Steps for Installing a SCSI VMScluster System

	Description	Reference
❶	Ensure proper grounding between enclosures.	Section C.6.1 and Section C.7.8
❷	Configure SCSI host IDs.	Section C.6.2
❸	Power up the system and verify devices.	Section C.6.3
❹	Set SCSI console parameters.	Section C.6.4
❺	Install the OpenVMS operating system.	Section C.6.5
❻	Configure additional systems.	Section C.6.6

C.6.1 Step 1: Meet SCSI Grounding Requirements

You must ensure that your electrical power distribution systems meet local requirements (for example, electrical codes) prior to installing your VMScluster system. If your configuration consists of two or more enclosures connected by a common SCSI interconnect, you must also ensure that the enclosures are properly grounded. Proper grounding is important for safety reasons and to ensure the proper functioning of the SCSI interconnect.

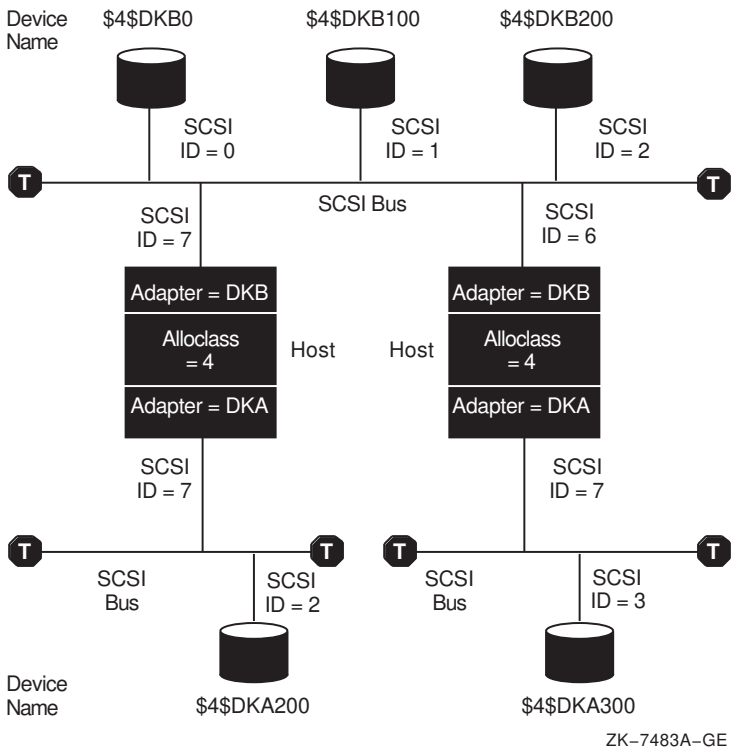
Electrical work should be done by a qualified professional. Section C.7.8 includes details of the grounding requirements for SCSI systems.

C.6.2 Step 2: Configure SCSI Node IDs

This section describes how to configure SCSI node and device IDs. SCSI IDs must be assigned separately for multiple-host SCSI buses and single-host SCSI buses.

Figure C–15 shows two hosts; each one is configured with a single-host SCSI bus and shares a multiple-host SCSI bus. (See Figure C–1 for the key to the symbols used in this figure.)

Figure C–15 Setting Allocation Classes for SCSI Access



The following sections describe how IDs are assigned in this type of multiple-host SCSI configuration. For more information about this topic, see *VMScluster Systems for OpenVMS*.

C.6.2.1 Configuring Device IDs on Multiple-Host SCSI Buses

When configuring multiple-host SCSI buses, adhere to the following rules:

- Set each host adapter on the multiple-host bus to a different ID. Start by assigning ID 7, then ID 6, and so on, using decreasing ID numbers.

  If a host has two multiple-host SCSI buses, allocate an ID to each SCSI adapter separately. There is no requirement that you set the adapters to the same ID, although using the same ID may simplify configuration management. (Section C.6.4 describes how to set host IDs for the internal adapter using SCSI console parameters.)

- When assigning IDs to devices and storage controllers connected to multiple-host SCSI buses, Digital recommends starting at ID 0 (zero), assigning the highest ID numbers to the disks that require the fastest I/O response time.

- Devices connected to a multiple-host SCSI bus must have the same name as viewed from each host. To achieve this, ensure that:

  – All hosts connected to a multiple-host SCSI bus are set to the same allocation class

  – All host adapters connected to a multiple-host SCSI bus have the same controller letter

C.6.2.2 Configuring Device IDs on Single-Host SCSI Buses

The following discussion applies to hosts that include both a single-host SCSI bus and a multiple-host SCSI bus.

In multiple-host SCSI configurations, device names generated by OpenVMS use the format $*allocation_class*$DKA300. You set the allocation class using the ALLOCLASS system parameter. OpenVMS generates the controller letter (for example, A, B, C, and so forth) at boot time by allocating a letter to each controller. The unit number (for example, 0, 100, 200, 300, and so forth) is derived from the SCSI device ID.

When configuring devices on single-host SCSI buses that are part of a multiple-host SCSI configuration, take care to ensure that the disks connected to the single-host SCSI buses have unique device names. Do this by assigning different IDs to devices connected to single-host SCSI buses with the same controller letter on systems that use the same allocation class. Note that the device names must be different, even though the bus is not shared.

For example, in Figure C–15, the two disks at the bottom of the picture are located on SCSI bus A of two systems that use the same allocation class. Therefore, they have been allocated different device IDs (in this case 2 and 3).

For a given allocation class, SCSI device type, and controller letter (in this example, ''4DKA''), there can be up to 8 devices in the cluster, one for each SCSI bus ID. To use all 8 IDs, it is necessary to configure a disk on one SCSI bus at the same ID as a processor on another bus. See Section C.7.5 for a discussion of the possible performance impact this can have.)

SCSI bus IDs can be effectively ''doubled up'' by configuring different SCSI device types at the same SCSI ID on different SCSI buses. For example, device types ''DK'' and ''MK'' could produce ''4DKA100'' and ''4MKA100''.

C.6.3 Step 3: Power Up and Verify SCSI Devices

After connecting the SCSI cables, power up the system. Enter a console SHOW DEVICE command to verify that all devices are visible on the SCSI interconnect.

If there is a SCSI ID conflict, the display may omit devices that are present, or it may include nonexistent devices. If the display is incorrect, then check the SCSI ID jumpers on devices, the automatic ID assignments provided by the StorageWorks shelves, and the console settings for host adapter and HSZ40 controller IDs. If changes are made, type INIT, then SHOW DEVICE again. If problems persist, check the SCSI cable lengths and termination.

The following is a sample output from a console SHOW DEVICE command. This system has one host SCSI adapter on a private SCSI bus (pka0), and two additional SCSI adapters (pkb0 and pkc0), each on separate, shared SCSI buses.

```
>>>SHOW DEVICE
dka0.0.0.6.0             DKA0 ❶                    RZ26L  442D
dka400.4.0.6.0           DKA400                    RRD43  2893
dkb100.1.0.11.0          DKB100                     RZ26  392A
dkb200.2.0.11.0          DKB200                    RZ26L  442D
dkc400.4.0.12.0          DKC400                    HSZ40   V25
dkc401.4.0.12.0          DKC401 ❷                  HSZ40   V25
dkc500.5.0.12.0          DKC500                    HSZ40   V25
dkc501.5.0.12.0          DKC501                    HSZ40   V25
dkc506.5.0.12.0          DKC506                    HSZ40   V25
dva0.0.0.0.1             DVA0
jkb700.7.0.11.0          JKB700 ❸                OpenVMS  V62
jkc700.7.0.12.0          JKC700                  OpenVMS  V62
mka300.3.0.6.0           MKA300 ❹                 TLZ06  0389
era0.0.0.2.1             ERA0          08-00-2B-3F-3A-B9
pka0.7.0.6.0             PKA0 ❺             SCSI Bus ID 7
pkb0.6.0.11.0            PKB0               SCSI Bus ID 6
pkc0.6.0.12.0            PKC0               SCSI Bus ID 6
```

The following list describes the device names in the preceding example:

❶ DK devices represent SCSI disks. Disks connected to the SCSI bus controlled by adapter PKA are given device names starting with the letters DKA. Disks on additional buses are named according to the host adapter name in a similar manner (DKB devices on adapter PKB, and so on).

The next character in the device name (0 in this case) represents the device's SCSI ID. Make sure that the SCSI ID for each device is unique for the SCSI bus to which it is connected.

❷ The last digit in the DK device name (1 in this case) represents the LUN number. The HSZ40 virtual DK device in this example is at SCSI ID 4, LUN 1. Note that some systems do not display devices that have nonzero LUNs.

❸ JK devices represent non-disk or non-tape devices on the SCSI interconnect. In this example, JK devices represent other processors on the SCSI interconnect that are running the OpenVMS operating system. If the other system is not running, these JK devices do not appear in the display. From this example, we can see that the other processor's adapters are at SCSI ID 7.

❹ MK devices represent SCSI tapes. The third character in this device's name is A, indicating that it is attached to adapter pka0, the private SCSI bus.

❺ PK devices represent the local SCSI adapters. The information in the rightmost column indicates this adapter's SCSI ID. Make sure this is different from the IDs used by other devices and host adapters on its bus.

The third character in the device name (in this example, a) is assigned by the system so that each adapter has a unique name on that system. The fourth character is always zero.

Note

Make sure that all host adapters attached to a SCSI interconnect have the same name; that is, the third character in the device names must be the same. OpenVMS configures the internal SCSI adapter as DKA, the first KZPAA adapter as DKB, and so on. For example, to ensure that device names are consistent on a shared SCSI bus, do not connect the internal adapter in one system and a KZPAA adapter in a second system to the same bus. For more information, see Section C.7.4.2.1.

C.6.4 Step 4: Show and Set SCSI Console Parameters

When creating a SCSI VMScluster system, you need to verify the settings of the console environment parameters shown in Table C–7 and, if necessary, reset their values according to your configuration requirements.

Table C–7 provides a brief description of SCSI console parameters. Refer to your system-specific documentation for complete information about setting these and other system parameters.

Note

If you need to modify any parameters, first change the parameter (using the appropriate console SET command), and then enter a console INIT command or press the Reset button to make the change effective.

Table C–7 SCSI Environment Parameters

Parameter	Description
bootdef_dev *device_name*	Specifies the default boot device to the system.
boot_osflags *root_ number,bootflag*	The boot_osflags variable contains information that is used by the operating system to determine optional aspects of a system bootstrap (for example, conversational bootstrap).
pk*0_disconnect	Allows the target to disconnect from the SCSI bus while the target acts on a command. When this parameter is set to 1, the target is allowed to disconnect from the SCSI bus while processing a command. When the parameter is set to 0, the target retains control of the SCSI bus while acting on a command.
pk*0_fast	Enables SCSI adapters to perform in Fast SCSI Mode. When this parameter is set to 1, the default speed is set to Fast Mode; when the parameter is 0, the default speed is Standard Mode.
pk*0_host_id	Sets the SCSI device ID of host adapters to a value between 0 and 7.
scsi_poll	Enables console polling on all SCSI interconnects when the system is halted.
control_scsi_term	Enables and disables the terminator on the integral SCSI interconnect at the system bulkhead (for some systems).

Before setting boot parameters, display the current settings of these parameters, as shown in the following examples:

Examples

1.
```
>>>SHOW *BOOT*

boot_osflags            10,0
boot_reset              OFF
bootdef_dev             dka200.2.0.6.0
>>>
```

   The first number in the boot_osflags parameter specifies the system root. (In this example, the first number is 10.) The boot_reset parameter controls the boot process. The default boot device is the device from which the OpenVMS operating system is loaded. Refer to the documentation for your specific system for additional booting information.

Note that you can identify multiple boot devices to the system. By doing so, you cause the system to search for a bootable device from the list of devices that you specify. The system then automatically boots from the first device on which it finds bootable system software. In addition, you can override the default boot device by specifying an alternative device name on the boot command line.

Typically, the default boot flags suit your environment. You can override the default boot flags by specifying boot flags dynamically on the boot command line with the -flags option.

2.
```
>>>SHOW *PK*

pka0_disconnect         1
pka0_fast               1
pka0_host_id            7
```

The pk*0_disconnect parameter determines whether or not a target is allowed to disconnect from the SCSI bus while it acts on a command. On a multiple-host SCSI bus, the pk*0_disconnect parameter *must* be set to 1, so that disconnects can occur.

The pk*0_fast parameter controls whether Fast SCSI devices on a SCSI controller perform in Standard or Fast Mode. When the parameter is set to 0, the default speed is set to standard mode; when the pk*0_fast parameter is set to 1, the default speed is set to Fast SCSI Mode. In this example, devices on SCSI controller pka0 are set to Fast SCSI Mode. This means that both Standard and Fast SCSI devices connected to this controller will automatically perform at the appropriate speed for the device (that is, in either Fast or Standard Mode).

The pk*0_host_id parameter assigns a bus node ID for the specified host adapter. In this example, pka0 is assigned a SCSI device ID of 7.

3.
```
>>>SHOW *POLL*
scsi_poll               ON
```

Enables or disables polling of SCSI devices while in console mode.

Set polling ON or OFF depending on the needs and environment of your site. When polling is enabled, the output of the SHOW DEVICE is always up-to-date. However, because polling can consume SCSI bus bandwidth (proportional to number of unused SCSI IDs), you might want to disable polling if one system on a multiple-host SCSI bus will be in console mode for an extended period of time.

Polling *must* be disabled during any hot plugging operations. For information on hot plugging in a SCSI VMScluster environment, see Section C.7.6.

4.
```
>>>SHOW *TERM*
control_scsi_term       external
```

Used on some systems (such as the AlphaStation 400) to enable or disable the SCSI terminator next to the external connector. Set the control_scsi_term parameter to external if a cable is attached to the bulkhead. Otherwise, set the parameter to internal.

C.6.5 Step 5: Install the OpenVMS Operating System

Refer to the OpenVMS Alpha or VAX upgrade and installation manual for information about installing the OpenVMS operating system. Perform the installation once for each system disk in the VMScluster system. In most configurations, there is a single system disk. Therefore, you need to perform this step once, using any system.

During the installation, when you are asked if the system is to be a cluster member, answer Yes. Then, complete the installation according to the guidelines provided in *VMScluster Systems for OpenVMS*.

C.6.6 Step 6: Configure Additional Systems

Use the CLUSTER_CONFIG command procedure to configure additional systems. Execute this procedure once for the second host that you have configured on the SCSI bus. (See Section C.7.1 for more information.)

C.7 Supplementary Information

The following sections provide supplementary technical detail and concepts about SCSI VMScluster systems.

C.7.1 Running the CLUSTER_CONFIG Command Procedure

You execute the CLUSTER_CONFIG.COM command procedure to set up and configure nodes in your VMScluster system.† Typically, the first computer is set up as a VMScluster system during the initial OpenVMS installation procedure (see Section C.6.5). The CLUSTER_CONFIG procedure is then used to configure additional nodes. However, if you originally installed OpenVMS without enabling clustering, the first time you run CLUSTER_CONFIG, the procedure converts the standalone system to a cluster system.

To configure additional nodes in a SCSI cluster, execute CLUSTER_CONFIG.COM for each additional node. Table C–8 describes the steps to configure additional SCSI nodes.

Table C–8 Steps for Installing Additional Nodes

	Procedure
❶	From the first node, run the CLUSTER_CONFIG.COM procedure and select the default option [1] for ADD.
❷	Answer Yes when CLUSTER_CONFIG.COM asks if you want to proceed.
❸	Supply the DECnet name and address of the node that you are adding to the existing single-node cluster.
❹	Confirm that this will be a node with a shared SCSI interconnect.
❺	Answer No when the procedure asks if this node will be a satellite.
❻	Configure the node to be a disk server if it will serve disks to other cluster members.
❼	Place the new node's system root on the default device offered.

(continued on next page)

† In OpenVMS Version 6.2, sites that choose to boot their VMScluster systems using the LANCP Utility rather than DECnet use the CLUSTER_CONFIG_LAN.COM procedure. See *OpenVMS Version 6.2 New Features Manual* for information about the use of this alternative procedure.

Table C–8 (Cont.) Steps for Installing Additional Nodes

	Procedure
❽	Select a system root for the new node. The first node uses SYS0. Take the default (SYS10 for the first additional node), or choose your own root numbering scheme. You can choose from SYS1 to SYS*n*, where *n* is hexadecimal FFFF.
❾	Select the default disk allocation class so that the new node in the cluster will use the same ALLOCLASS as the first node.
❿	Confirm whether or not there is a quorum disk.
⓫	Answer the questions about the sizes of the page file and swap file.
⓬	When CLUSTER_CONFIG.COM completes, boot the new node from the new system root. For example, for SYSFF on disk DKA200, enter the following command: `BOOT -FL FF,0 DKA200` On the BOOT command, you can use the following flags: • -FL indicates boot flags • FF is the new system root • 0 means there are no special boot requirements, such as conversational boot

Example C–1 shows how to run the CLUSTER_CONFIG.COM procedure to set up an additional node in a SCSI cluster.

Example C–1 Adding a Node to a SCSI Cluster

```
$ @SYS$MANAGER:CLUSTER_CONFIG

          Cluster Configuration Procedure

    Use CLUSTER_CONFIG.COM to set up or change a VMScluster configuration.
    To ensure that you have the required privileges, invoke this procedure
    from the system manager's account.

    Enter ? for help at any prompt.

           1. ADD a node to a cluster.
           2. REMOVE a node from the cluster.
           3. CHANGE a cluster member's characteristics.
           4. CREATE a duplicate system disk for CLU21.
           5. EXIT from this procedure.

    Enter choice [1]:

    The ADD function adds a new node to a cluster.

    If the node being added is a voting member, EXPECTED_VOTES in
    every cluster member's MODPARAMS.DAT must be adjusted, and the
    cluster must be rebooted.

    WARNING - If this cluster is running with multiple system disks and
              if common system files will be used, please, do not
              proceed unless you have defined appropriate logical
              names for cluster common files in SYLOGICALS.COM.
              For instructions, refer to the VMScluster Systems for
              OpenVMS manual.

              Do you want to continue [N]? y
```

(continued on next page)

Example C–1 (Cont.) Adding a Node to a SCSI Cluster

```
    If the new node is a satellite, the network databases on CLU21 are
    updated. The network databases on all other cluster members must be
    updated.

    For instructions, refer to the VMScluster Systems for OpenVMS manual.

What is the node's DECnet node name? SATURN
What is the node's DECnet node address? 7.77
Is SATURN to be a clustered node with a shared SCSI bus (Y/N)? y
Will SATURN be a satellite [Y]? N
Will SATURN be a boot server [Y]?

    This procedure will now ask you for the device name of SATURN's system root.
    The default device name (DISK$BIG_X5T5:) is the logical volume name of
    SYS$SYSDEVICE:.

What is the device name for SATURN's system root [DISK$BIG_X5T5:]?
What is the name of SATURN's system root [SYS10]? SYS2
    Creating directory tree SYS2 ...
    System root SYS2 created

    NOTE:
        All nodes on the same SCSI bus must be members of the same cluster
        and must all have the same non-zero disk allocation class or each
        will have a different name for the same disk and data corruption
        will result.

Enter a value for SATURN's ALLOCLASS parameter [7]:
Does this cluster contain a quorum disk [N]?
Updating network database...
Size of pagefile for SATURN [10000 blocks]?
   .
   .
   .
```

C.7.2 Error Reports and OPCOM Messages in Multiple-Host SCSI Environments

Certain common operations, such as booting or shutting down a host on a multiple-host SCSI bus, can cause other hosts on the SCSI bus to experience errors. In addition, certain errors that are unusual in a single-host SCSI configuration may occur more frequently on a multiple-host SCSI bus.

These errors are transient errors that OpenVMS detects, reports, and recovers from without losing data or affecting applications that are running. This section describes the conditions that generate these errors and the messages that are displayed on the operator console and entered into the error log.

C.7.2.1 SCSI Bus Resets

When a host connected to a SCSI bus first starts, either by being turned on or by rebooting, it does not know the state of the SCSI bus and the devices on it. The ANSI SCSI–2 Standard provides a method called BUS RESET to force the bus and its devices into a known state. A host typically asserts a RESET signal one or more times on each of its SCSI buses when it first starts up and when it shuts down. While this is a normal action on the part of the host asserting RESET, other hosts consider this RESET signal an error, because RESET requires that the hosts abort and restart all I/O operations that are in progress.

A host may also reset the bus in the midst of normal operation if it detects a problem that it cannot correct in any other way. These kinds of resets are uncommon but occur most frequently when something on the bus is disturbed. For example, an attempt to hot plug a SCSI device while the device is still active (see Section C.7.6) or halting one of the hosts with CTRL/P can cause a condition that forces one or more hosts to issue a bus reset.

C.7.2.2 SCSI Timeouts

When a host exchanges data with a device on the SCSI bus, there are several different points where the host must wait for the device or the SCSI adapter to react. In an OpenVMS system, the host is allowed to do other work while it is waiting, but a timer is started to make sure that it does not wait too long. If the timer expires without a response from the SCSI device or adapter, this is called a timeout.

There are three kinds of timeouts:

- Disconnect timeout—The device accepted a command from the host and disconnected from the bus while it processed the command but never reconnected to the bus to finish the transaction. This error happens most frequently when the bus is very busy. See Section C.7.5 for more information. The disconnect timeout period varies with the device, but for most disks, it is about 20 seconds.
- Selection timeout—The host tried to send a command to a device on the SCSI bus, but the device did not respond. This condition might happen if the device did not exist, or if it were removed from the bus or powered down, for example. (This failure is not more likely with a multi-initiator system; it is mentioned here for completeness.) The selection timeout period is about 0.25 seconds.
- Interrupt timeout—The host expected the adapter to respond for any other reason, but it did not respond. This error is usually an indication of a busy SCSI bus. It is more common if you have initiator unit numbers set low (0 or 1) rather than high (6 or 7). The interrupt timeout period is about 4 seconds.

Timeout errors are not inevitable on SCSI VMScluster systems. However, they are more frequent on SCSI buses with heavy traffic and those with two initiators. They do not necessarily indicate a hardware or software problem. If they are logged frequently, you should consider ways to reduce the load on the SCSI bus (for example adding an additional bus).

C.7.2.3 Mount Verify

Mount verify is a condition declared by a host about a device. The host declares this condition in response to a number of possible transient errors, including bus resets and timeouts. When a device is in the mount verify state, the host suspends normal I/O to it until the host can determine that the correct device is there, and that the device is accessible. Mount verify processing then retries outstanding I/Os in a way that insures that the correct data is written or read. Application programs are unaware that a mount verify condition has occurred as long as the mount verify completes.

If the host cannot access the correct device within a certain amount of time, it declares a mount verify timeout, and application programs are notified that the device is unavailable. Manual intervention is required to restore a device to service after the host has declared a mount verify timeout. A mount verify timeout usually means that the error is not transient. The system manager can choose the timeout period for mount verify; the default is one hour.

C.7.2.4 Shadow Volume Processing

Shadow volume processing is a process similar to mount verify, but it is for shadow set members. An error on one member of a shadow set places the set into the volume processing state, which blocks I/O while OpenVMS attempts to regain access to the member. If access is regained before shadow volume processing times out, then the outstanding I/Os are reissued and the shadow set returns to normal operation. If a timeout occurs, then the failed member is removed from the set. The system manager can select one timeout value for the system disk shadow set, and one for application shadow sets. The default value for both timeouts is 20 seconds.

Note

The SCSI disconnect timeout and the default shadow volume processing timeout are the same. If the SCSI bus is heavily utilized so that disconnect timeouts may occur, it may be desirable to increase the value of the shadow volume processing timeout. (A recommended value is 60 seconds.) This may prevent shadow set members from being expelled when they experience disconnect timeout errors.

C.7.2.5 Expected OPCOM Messages in Multiple-Host SCSI Environments

When a bus reset occurs, an OPCOM message is displayed as each mounted disk enters and exits mount verification or shadow volume processing.

When an I/O to a drive experiences a timeout error, an OPCOM message is displayed as that drive enters and exits mount verification or shadow volume processing.

If a quorum disk on the shared SCSI bus experiences either of these errors, then additional OPCOM messages may appear, indicating that the connection to the quorum disk has been lost and regained.

C.7.2.6 Error-Log Basics

In the OpenVMS system, the Error Log utility allows device drivers to save information about unusual conditions that they encounter. In the past, most of these unusual conditions have happened as a result of errors such as hardware failures, software failures, or transient conditions (like loose cables, for example).

If you type the DCL command SHOW ERROR, the system displays a summary of the errors that have been logged since the last time the system booted. For example:

```
$ SHOW ERROR

Device                            Error Count
SALT$PKB0:                              6
$1$DKB500:                             10
PEA0:                                   1
SALT$PKA0:                              9
$1$DKA0:                                0
```

In this case, 6 errors have been logged against host SALT's SCSI port B (PKB0), 10 have been logged against disk 1DKB500, and so forth.

To see the details of these errors, you can use the command ANALYZE/ERROR /SINCE=DD-MMM-YYYY:HH:MM:SS at the DCL prompt. The output from this command displays a list of error-log entries with information similar to the following:

```
******************************* ENTRY    2337. *******************************
 ERROR SEQUENCE 6.                              LOGGED ON:  CPU_TYPE 00000002
 DATE/TIME 29-MAY-1995 16:31:19.79                           SYS_TYPE 0000000D

<identification information>

       ERROR TYPE            03
                                        COMMAND TRANSMISSION FAILURE
       SCSI ID               01
                                        SCSI ID = 1.
       SCSI LUN              00
                                        SCSI LUN = 0.
       SCSI SUBLUN           00
                                        SCSI SUBLUN = 0.
       PORT STATUS     00000E32
                                        %SYSTEM-E-RETRY, RETRY OPERATION

<additional information>
```

For this discussion, the key elements are the ‘‘ERROR TYPE’’ and, in some instances, the ‘‘PORT STATUS’’ fields. In this example, the ERROR TYPE is ‘‘03, COMMAND TRANSMISSION FAILURE’’, and the PORT STATUS is ‘‘00000E32, SYSTEM-E-RETRY’’.

C.7.2.7 Error-Log Entries in Multiple-Host SCSI Environments

The error-log entries listed in this section are likely to be logged in a multiple-host SCSI configuration, and you usually do not need to be concerned about them. You should, however, examine any error-log entries for messages other than those listed in this section.

- ERROR TYPE 0007, BUS RESET DETECTED

  Occurs when the other system asserts the SCSI bus reset signal. This happens when a:

  - System’s power-up self-test runs
  - Console INIT command is executed
  - EISA Configuration utility (ECU) is run
  - Console BOOT command is executed (several resets occur in this case)
  - System shutdown completes
  - System detects a problem with an adapter or a SCSI bus (for example, an interrupt timeout)

  This error causes all mounted disks to enter mount verification.

- ERROR TYPE 05, EXTENDED SENSE DATA RECEIVED

  When a SCSI bus is reset, an initiator must get ‘‘sense data’’ from each device. When the initiator gets this data, an ‘‘EXTENDED SENSE DATA RECEIVED’’ error is logged. This is expected behavior.

- ERROR TYPE 03, COMMAND TRANSMISSION FAILURE
  PORT STATUS E32, SYSTEM-E-RETRY

Occasionally, one host may send a command to a disk while the disk is exchanging error information with the other host. Many disks respond with a SCSI ''BUSY'' code. The OpenVMS system responds to a SCSI BUSY code by logging this error and retrying the operation. You are most likely to see this error when the bus has been reset recently. This error does not always happen near resets, but when it does, the error is expected and unavoidable.

- ERROR TYPE 204, TIMEOUT

  An interrupt timeout has occurred (see Section C.7.2.2). The disk is put into mount verify when this error occurs.

- ERROR TYPE 104, TIMEOUT

  A selection timeout has occurred (see Section C.7.2.2). The disk is put into mount verify when this error occurs.

C.7.3 Restrictions and Known Problems

The current release of VMSclusters has the following restrictions when multiple hosts are configured on the same SCSI bus:

1. A node's access to a disk will not failover from a direct SCSI path to an MSCP served path. This is not expected to be a significant limitation, since most of the failures that cause a SCSI disk to become inaccessible to one node on the SCSI bus impacts all the nodes on the SCSI bus. Thus, when a failure occurs, the served path to the disk tends to fail at the same time that the direct path fails.

   Conversely, a node's access to a disk will not failover from an MSCP served path to a direct SCSI path. Normally, this type of failover is not a consideration, because when OpenVMS discovers both a direct and a served path, it chooses the direct path permanently. It is necessary, however, to avoid situations in which the MSCP served path becomes available first, and is selected by OpenVMS before the direct path becomes available. You must avoid this by observing the following rules:

   - A node that has a direct path to a SCSI system disk must boot the disk directly from the SCSI port, not over the LAN.
   - If a node is running the MSCP server, then a SCSI disk must not be added to the multiple-host SCSI bus after the node boots. This is necessary to prevent the second node on the SCSI bus from seeing the served path to the new disk and configuring it, thereby precluding the second node from configuring its direct path.

2. The SYS$DEVICE_SCAN system service (and the F$DEVICE lexical function that calls it) can be executed repeatedly to obtain a list of devices on the system. If this is done on a system with a multiple-host SCSI bus, and the other system is running the MSCP server, then each device on the multiple-host SCSI bus is reported twice by SYS$DEVICE_SCAN (or F$DEVICE).

   The reason for this is that each device on the multiple-host SCSI bus has two UCBs, one for the direct SCSI path, and one for the MSCP served path. (The MSCP served path is not used.)

   Programs that use SYS$DEVICE_SCAN or F$DEVICE to search the IO database may need to be modified to check for, and ignore, duplicate device names.

3. If a system on a multiple-host SCSI bus boots a disk that is served to it over the LAN, and the other system on the SCSI bus is running the MSCP server, then each device on the multiple-host SCSI bus is reported twice by the DCL SHOW DEVICE command. This not known to result in any other adverse effects.
4. Abruptly halting a system on a multiple-host SCSI bus (by typing CTRL/P on the console, for example) may leave the SCSI adapter in a state that can interfere with the operation of the other host on the bus. It is recommended that you either initialize, boot, or continue an abruptly halted system as soon as possible after it has been halted.
5. All I/O to a disk drive must be stopped while its microcode is updated. This typically requires more precautions in a multiple-host environment than are needed in a single-host environment. Refer to Section C.7.6.3 for the necessary procedures.
6. The EISA Configuration Utility (ECU) causes a large number of SCSI bus resets. These resets cause the other system on the SCSI bus to pause while its I/O subsystem recovers. It is suggested (though not required) that both systems on a shared SCSI bus be shut down when the ECU is run.

The current release of VMSclusters also places one new restriction on the SCSI quorum disk, whether the disk is located on a single-host SCSI bus or a multiple-host SCSI bus: the SCSI quorum disk must support Tagged Command Queueing. This is required because of the special handling that quorum I/O receives in the OpenVMS SCSI drivers.

This restriction is not expected to be significant, because all disks on a multiple-host SCSI bus must support Tagged Command Queueing (see Section C.7.7), and because quorum disks are normally not used on single-host buses.

C.7.4 Troubleshooting

The following sections describe troubleshooting tips for solving common problems in a VMScluster system using a SCSI interconnect.

C.7.4.1 Termination Problems

Verify that two terminators are on every SCSI interconnect (one at each end of the interconnect). The BA350 enclosure, the DWZZA, and the KZPAA adapter have internal terminators that are not visible externally (see Section C.4.4.)

C.7.4.2 Booting or Mounting Failures Caused by Incorrect Configurations

OpenVMS automatically detects configuration errors described in this section and prevents the possibility of data loss that could result from such configuration errors, either by bugchecking or by refusing to mount a disk.

C.7.4.2.1 Bugchecks During the Bootstrap Process There are three types of configuration error that can cause a bugcheck (the bugcheck code is: ''VAXCLUSTER, Error detected by VMScluster software'' during booting. These are described in this section.

When OpenVMS boots, it determines which devices are present on the SCSI bus by sending an inquiry command to every SCSI ID. When a device receives the inquiry, it indicates its presence by returning data that indicates whether it is a disk, tape, or processor.

Some processor devices (host adapters) answer the inquiry without assistance from the operating system; others require that the operating system be running. The adapters supported in VMScluster systems require the operating system to be running. These adapters, with the aid of OpenVMS, pass information in their response to the inquiry that allows the recipient to detect the following configuration errors:

- Different controller device names on the same SCSI bus

  The OpenVMS device name of each adapter on the SCSI bus must be identical (all named pkc0, for example), or the VMScluster software cannot coordinate the host's accesses to storage (see Section C.6.2 and Section C.6.3).

  OpenVMS can check this automatically, because it sends the controller letter in the inquiry response. A booting system receives this response, and it compares the remote controller letter with the local controller letter. If a mismatch is detected, then an OPCOM message is printed, and the system stops with an VAXCLUSTER bugcheck to prevent the possibility of data loss. See the description of the NOMATCH error in either the Help Message utility or in the *OpenVMS Version 6.2 New Features Manual*. (To use the Help Message utility for NOMATCH, enter HELP/MESSAGE NOMATCH at the DCL prompt.)

- Different, or zero allocation class values.

  Each host on the SCSI bus must have the same non-zero disk allocation class value, or the VMScluster software cannot coordinate the host's accesses to storage (see Section C.6.2 and Section C.6.3. The disk allocation class value is controlled by the ALLOCLASS SYSGEN parameter.

  OpenVMS is able to automatically check this, because it sends the ALLOCLASS value in the inquiry response. A booting system receives this response, and compares the remote ALLOCLASS value with the local ALLOCLASS value. If a mismatch or a zero value is detected, then an OPCOM message is printed, and the system stops with a VAXCLUSTER bugcheck, in order to prevent the possibility of data loss. See the description of the ALLODIFF and ALLOZERO errors in either the Help Message utility or in the *OpenVMS Version 6.2 New Features Manual*.

- Unsupported processors

  Finally, there may be processors on the SCSI bus that are not running OpenVMS or that do not return the controller name or allocation class information needed to validate the configuration. If a booting system receives an inquiry response and the response does not contain the special OpenVMS configuration information, then an OPCOM message is printed and an VAXCLUSTER bugcheck occurs. See the description of the CPUNOTSUP error in either the Help Message utility or in the *OpenVMS Version 6.2 New Features Manual*.

  (If your system requires the presence of a VMScluster processor device on a SCSI bus, then refer to the CPUNOTSUP message description in either the Help Message utility or in the *OpenVMS Version 6.2 New Features Manual* for instructions on the use of a special SYSGEN parameter for this purpose.)

Hint

The OPCOM error code that is printed for each of the failures described above is preserved in R8 of the VAXCLUSTER bugcheck. You can examine register R8 to quickly determine the cause of the error. For example:

```
$ ANAL/CRASH SYSDUMP.DMP
SDA> examine @r8;50
2D462D47 49464E4F 43415453 250A0D4B  K..%STACONFIG-F-     00020430
4C412065 6854202C 46464944 4F4C4C41  ALLODIFF, The AL     00020440
6574656D 61726170 20535341 4C434F4C  LOCLASS paramete     00020450
20656874 20726F66 2065756C 61762072  r value for the      00020460
00000000 6E6F2072 6F737365 636F7270  processor on....     00020470
```

C.7.4.2.2 Mount Failures There are two types of configuration error that can cause a disk to fail to mount. These are described in this section.

First, when a system boots from a disk on the shared SCSI bus, it may fail to mount the system disk. This happens if there is another system on the SCSI bus that is already booted, and the other system is using a different device name for the system disk. (Two systems will disagree about the name of a device on the shared bus if their controller names or allocation classes are mis-configured, as described in the previous section.) If the system does not execute one of the bugchecks described in the previous section first, then the following error message is displayed on the console:

```
%SYSINIT-E- error when mounting system device, retrying..., status = 007280B4
```

The decoded representation of this status is:

```
VOLALRMNT,  another volume of same label already mounted
```

This error indicates that the system disk is already mounted in what appears to be another drive in the VMScluster system, so it is not mounted again. Solve this problem by checking the controller letters and allocation class values for each node on the shared SCSI bus.

Second, SCSI disks on a shared SCSI bus will fail to mount on both systems unless the disk supports Tagged Command Queueing (TCQ). This is because TCQ provides a command ordering guarantee that is required during VMScluster state transitions.

OpenVMS determines that another processor is present on the SCSI bus during autoconfiguration, using the mechanism described in Section C.7.4.2.1. The existence of another host on a SCSI bus is recorded and preserved until the system reboots.

This information is used whenever an attempt is made to mount a non-TCQ device. If the device is on a multiple-host bus, the mount attempt fails and returns the following message:

```
%MOUNT-F-DRVERR, fatal drive error.
```

If the drive is intended to be mounted by multiple hosts on the same SCSI bus, then it must be replaced with one that supports TCQ.

Note that the first processor to boot on a multiple-host SCSI bus does not receive an inquiry response from the other hosts, because the other hosts are not yet running OpenVMS. Thus, the first system to boot is unaware that the bus has multiple hosts, and it allows non-TCQ drives to be mounted. The other hosts on the SCSI bus detect the first host, however, and they are prevented from mounting the device. If two processors boot simultaneously, it is possible that they will detect each other, in which case neither is allowed to mount non-TCQ drives on the shared bus.

C.7.4.3 Grounding

Having excessive ground offset voltages or exceeding the maximum SCSI interconnect length can cause system failures or degradation in performance. See Section C.7.8 for more information about SCSI grounding requirements.

C.7.4.4 Interconnect Lengths

Adequate signal integrity depends on strict adherence to SCSI bus lengths. Failure to follow the bus length recommendations can result in problems (for example, intermittent errors) that are difficult to diagnose. See Section C.4.3 for information on SCSI bus lengths.

C.7.5 SCSI Arbitration Considerations

Only one initiator (typically, a host system) or target (typically, a peripheral device) can control the SCSI bus at any one time. In a computing environment where multiple targets frequently contend for access to the SCSI bus, you could experience throughput issues for some of these targets. This section discusses control of the SCSI bus, how that control can affect your computing environment, and what you can do to achieve the most desirable results.

Control of the SCSI bus changes continually. When an initiator gives a command (such as READ) to a SCSI target, the target typically disconnects from the SCSI bus while it acts on the command, allowing other targets or initiators to use the bus. When the target is ready to respond to the command, it must regain control of the SCSI bus. Similarly, when an initiator wishes to send a command to a target, it must gain control of the SCSI bus.

If multiple targets and initiators want control of the bus simultaneously, bus ownership is determined by a process called arbitration, defined by the SCSI Standard. The default arbitration rule is simple: control of the bus is given to the requesting initiator or target that has the highest unit number.

The following sections discuss some of the implications of arbitration and how you can respond to arbitration situations that affect your environment.

C.7.5.1 Arbitration Issues in Multi-Disk Environments

When the bus is not very busy, and bus contention is uncommon, the simple arbitration scheme is adequate to perform I/O requests for all devices on the system. However, as initiators make more and more frequent I/O requests, contention for the bus becomes more and more common. Consequently, targets with lower ID numbers begin to perform poorly, because they are frequently blocked from completing their I/O requests by other users of the bus (in particular, targets with the highest ID numbers). If the bus is sufficiently busy, low-numbered targets may never complete their requests. This situation is most likely to occur on systems with more than one initiator, because more commands can be outstanding at the same time.

The OpenVMS system attempts to prevent low-numbered targets from being completely blocked by monitoring the amount of time an I/O request takes. If the request is not completed within a certain period, the OpenVMS system stops sending new requests until the tardy I/Os complete. While this algorithm does not ensure that all targets get equal access to the bus, it does prevent low-numbered targets from being totally blocked.

C.7.5.2 Solutions for Resolving Arbitration Problems

If you find that some of your disks are not being serviced quickly enough during periods of heavy I/O, try some or all of the following, as appropriate for your site:

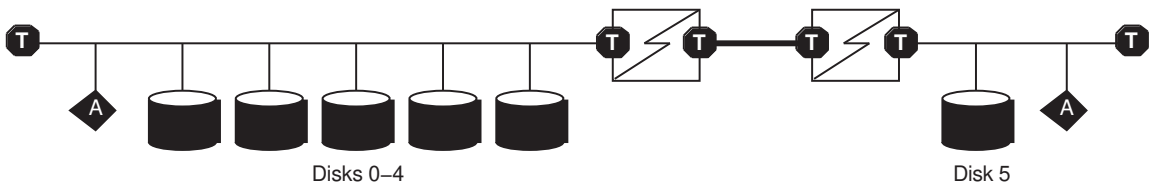
- Assign the highest ID numbers to those disks that require the fastest response time.
- Spread disks across more SCSI buses.
- Keep disks that need to be accessed only by a single host (for example, page and swap disks) on a nonshared SCSI bus.

Another method that might provide for more equal servicing of lower and higher ID disks is to set the host IDs to the lowest numbers (0 and 1) rather than the highest. When you use this method, the host cannot gain control of the bus to send new commands as long as any disk, including those with the lowest IDs, need the bus. Although this option is available to improve fairness under some circumstances, Digital considers this configuration to be less desirable in most instances, for the following reasons:

- It can result in lower total throughput.
- It can result in timeout conditions if a command cannot be sent within a few seconds.
- It can cause physical configuration difficulties. For example, StorageWorks shelves such as the BA350 have no slot to hold a disk with ID 7, but they do have a slot for a disk with ID 0. If you change the host to ID 0, you must remove a disk from slot 0 in the BA350, but you cannot move the disk to ID 7. If you have two hosts with IDs 0 and 1, you cannot use slot 0 or 1 in the BA350. (Note, however, that you *can* have a disk with ID 7 in a BA353.)

C.7.5.3 Arbitration and Bus Isolators

Any active device, such as a DWZZA, that connects bus segments introduces small delays as signals pass through the device from one segment to another. Under some circumstances, these delays can be another cause of unfair arbitration. For example, consider the following configuration, which could result in disk servicing problems (starvation) under heavy work loads:



ZK–7913A–GE

Although disk 5 has the highest ID number, there are some circumstances under which disk 5 has the lowest access to the bus. This can occur after one of the lower-numbered disks has gained control of the bus and then completed the operation for which control of the bus was needed. At this point, disk 5 does not recognize that the bus is free and might wait before trying to arbitrate for control

of the bus. As a result, one of the lower-numbered disks, having become aware of the free bus and then submitting a request for the bus, will gain control of the bus.

If you see this type of problem, the following suggestions can help you reduce its severity:

- Try to place all disks on the same bus segment.
- If placing all disks on the same bus segment is not possible (for example if you have both some RZ28 disks by themselves and an HSZ40), try to use a configuration that has only one isolator between any pair of disks.
- If your configuration requires two isolators between a pair of disks (for example, to meet distance requirements), try to balance the number of disks on each bus segment.
- Follow the suggestions in Section C.7.5.2 to reduce the total traffic on the logical bus.

C.7.6 Removal and Insertion of SCSI Devices While the VMScluster System is Operating

With proper procedures, certain SCSI devices can be removed from or inserted onto an active SCSI bus without disrupting the on-going operation of the bus. This capability is referred to as **hot plugging**. Hot plugging can allow a suitably configured VMScluster system to continue to run while a failed component is replaced. Without hot plugging, it is necessary to make the SCSI bus inactive and remove power from all the devices on the SCSI bus before any device is removed from it or inserted onto it.

In a SCSI VMScluster system, hot plugging requires that all devices on the bus have certain electrical characteristics and be configured appropriately on the SCSI bus. Successful hot plugging also depends on strict adherence to the procedures described in this section. These procedures ensure that the hot-plugged device is inactive and that active bus signals are not disturbed.

Hot Plugging for SCSI Buses Behind a Storage Controller

This section describes hot-plugging procedures for devices that are on the same SCSI bus as the host that is running OpenVMS. The procedures are different for SCSI buses that are behind a storage controller, such as the HSZ40. Refer to the storage controller documentation for the procedures to hot plug devices that they control.

C.7.6.1 Terminology for Describing Hot Plugging

The terms shown in bold in this section are used in the discussion of hot plugging rules and procedures.

- A SCSI bus **segment** consists of two terminators, the electrical path forming continuity between them, and possibly, some attached stubs. Bus segments may be connected together by bus isolators (for example, DWZZA), to form a **logical SCSI bus** or just **SCSI bus**.
- There are two types of connections on a segment: **bussing connections**, which break the path between two terminators, and **stubbing connections**, which disconnect all or part of a stub.

- A device is **active** on the SCSI bus when it is asserting one or more of the bus signals. A device is **inactive** when it is not asserting any bus signals.

  The segment attached to a bus isolator is inactive when all devices on that segment, except possibly the bus isolator, are inactive.

- A port on a bus isolator has **proper termination** when it is attached to a segment that is terminated at both ends and has TERMPWR in compliance with SCSI–2 requirements.

C.7.6.2 Rules for Hot Plugging

The following rules must be followed when planning for and performing hot plugging:

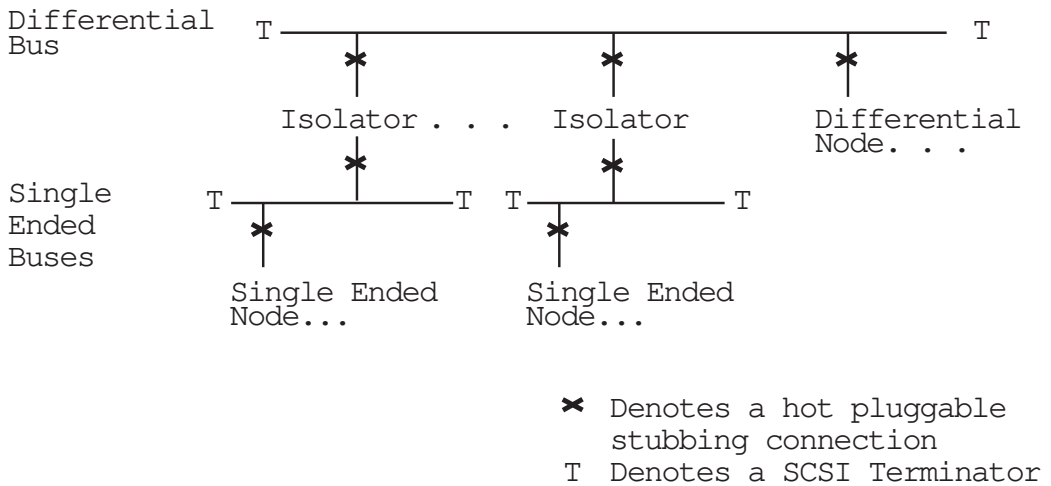
- The device to be hot plugged, and all other devices on the same segment, shall meet the electrical requirements described in Annex A, Section A.4, of the SCSI-3 Parallel Interface (SPI) Standard, working draft X3T10/855D.[1] The SPI document places requirements on the receivers and terminators on the segment where the hot plugging is being performed, and on the transceivers, TERMPWR, termination, and power/ground/signal sequencing, of the device that is being hot plugged.

  All the devices in Table C–2 meet these requirements, except the DWZZA. The DWZZA's transceivers do not meet the requirements for glitch-free power on/off. The rules and procedures in this section have been adjusted to compensate for this fact.

- Hot plugging shall occur only at a stubbing connection.

  This implies that a hot-plugged device shall make only one connection to the SCSI bus, the device shall not provide termination for the SCSI bus, and the device's connection shall not exceed the maximum stub length, as shown in Figure C–3. An example of a SCSI bus topology showing the valid hot plugging connections is illustrated in Figure C–16.

Figure C–16 SCSI Bus Topology



[1] Reference to this draft standard is necessary because the SCSI–2 standard does not adequately specify the requirements for hot plugging.

- Precautions shall be used to ensure that Electrostatic Discharge (ESD) does not damage devices or disrupt active signals on the SCSI bus. These precautions shall be taken during the process of disconnecting and connecting, as well as during the time that SCSI bus conductors are exposed.
- Precautions shall be used to ensure that ground offset voltages do not pose a safety hazard and will not interfere with SCSI bus signaling, especially in single-ended configurations. The procedures for measuring and eliminating ground offset voltages are described in Section C.7.8.
- The device that is hot plugged shall be inactive during the disconnection and connection operations. Otherwise, the SCSI bus may become hung.[1]

Note

Ideally, a device will also be inactive whenever its power is removed, for the same reason.

The procedures for ensuring that a device is inactive are described in Section C.7.6.3.

- A quorum disk shall not be hot plugged. This is because there is no mechanism for stopping the I/O to a quorum disk and because the replacement disk will not contain the correct quorum file.

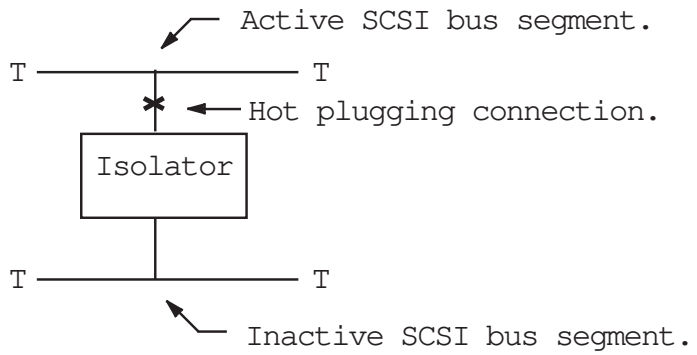
  The VMScluster system must be reconfigured to remove a device as a quorum disk before that device is removed from the bus. The procedure for accomplishing this is described in *VMScluster Systems for OpenVMS* (Section 8.3.3 in the OpenVMS Version 6.1 edition).

  An alternate method for increasing the availability of the quorum disk is to use an HSZ40 mirror set as the quorum disk. This would allow a failed member to be replaced while maintaining the quorum disk functionality.
- Disks shall be logically dismounted before removing or replacing them in a hot-plug operation. This is required to ensure that the disk is inactive and to ensure the integrity of the file system.
- The DWZZA shall be powered up when it is inserted onto an active SCSI bus. The DWZZA should remain powered up at all times while it is attached to the active SCSI bus. This is because the DWZZA can disrupt the operation of the attached segments when it is powering up or down.
- The segment attached to a bus isolator shall be maintained in the inactive state whenever the other port on the bus isolator is improperly terminated. This is required because an improperly terminated bus isolator port may pass erroneous signals to the other port.

  Thus, for a particular hot-plugging operation, one of the segments attached to a bus isolator shall be designated as the (potentially) active segment, and the other shall be maintained in the inactive state, as illustrated in Figure C–17. The procedures for ensuring that a segment is inactive are described in Section C.7.6.3.

[1] OpenVMS will eventually detect a hung bus and reset it, but this may be temporarily disruptive to VMScluster operations.

Figure C–17 Hot Plugging a Bus Isolator



Note that although a bus isolator may have more than one stubbing connection and thus be capable of hot plugging on each of them, only one segment can be the active segment for any particular hot-plugging operation.

- Precautions shall be taken to ensure that the only electrical conductor that contacts a connector pin is its mate. These precautions must be taken during the process of disconnecting and connecting as well as during the time the connector is disconnected.
- Devices shall be replaced with devices of the same type. That is, if any system in the VMScluster configures a SCSI ID as a ''DK'' or ''MK'' device, then that SCSI ID shall contain only ''DK'' or ''MK'' devices, respectively, for as long as that VMScluster member is running.

  Different implementations of the same device type may be substituted (for example, an RZ26L may be replaced with an RZ28B). Note that the system will not recognize the change in device type until an attempt is made to mount the new device. Also, note that host-based shadowing continues to require that all members of a shadow set be the same device type.
- SCSI IDs that are empty when a system boots shall remain empty as long as that system is running. This rule only applies if there are multiple processors on the SCSI bus and the MSCP server is loaded on any of them. (The MSCP server is loaded when the system parameter MSCP_LOAD equals 1).

  This is required to ensure that nodes on the SCSI bus use their direct path to the disk, rather than the served path. When the new device is configured on a system (using SYSMAN IO commands), that system serves it to the second system on the shared SCSI bus. The second system automatically configures the new device via the MSCP served path. Once this occurs, the second system will be unable to use its direct SCSI path to the new device, because failover from an MSCP served path to a direct SCSI path is not implemented.

C.7.6.3 Procedures for Ensuring That a Device or Segment Is Inactive

Use the following procedures to ensure that a device or a segment is inactive:

- To ensure that a disk is inactive:
  1. Dismount the disk on all members of the VMScluster system.
  2. Ensure that any I/O that can occur to a dismounted disk is stopped, for example:
     - Disable the disk as a quorum disk

- Allocate the disk (using the DCL ALLOCATE command) to block further mount or initialization attempts
- Disable console polling by all halted hosts on the logical SCSI bus (by setting the console variable SCSI_POLL to OFF, and entering the INIT command)
- Ensure that no host on the logical SCSI bus is executing power-up or initialization self-tests, booting, or configuring the SCSI bus (using SYSMAN IO commands).

- To ensure that an HSZ40 controller is inactive:
  1. Dismount all of the HSZ40 virtual disks on all members of the VMScluster system.
  2. Shut down the controller, following the procedures in the *HS Family of Array Controllers User's Guide*.
  3. Power down the HSZ40, if desired.
- To ensure that a host adapter is inactive:
  1. Halt the system.
  2. Power down the system, or set the console variable SCSI_POLL to OFF and then enter the INIT command on the halted system. This ensures that the system will not poll, or respond to polls.
- To ensure that a segment is inactive, follow the procedure described above for every device on the segment.

C.7.6.4 Procedure for Hot Plugging StorageWorks SBB Disks

To remove an SBB disk from an active SCSI bus, use the following procedure:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance.
2. Follow the procedure in Section C.7.6.3 to make the disk inactive.
3. Squeeze the clips on the side of the SBB, and slide the disk out of the StorageWorks shelf.

To insert an SBB disk onto an active SCSI bus, use the following procedure:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance.
2. Ensure that the SCSI ID associated with the device (either by jumpers or by the slot in the StorageWorks shelf) conforms to the following:
   - The SCSI ID is unique for the logical SCSI bus
   - The SCSI ID is already configured as a ''DK'' device on all of the following:
     - Any member of the VMScluster system that already has that ID configured
     - Any OpenVMS processor on the same SCSI bus that is running the MSCP server

3. Slide the SBB into the StorageWorks shelf.
4. Configure the disk on VMScluster members, if required, using SYSMAN IO commands.

C.7.6.5 Procedure for Hot Plugging HSZ40s

To remove an HSZ40 controller from an active SCSI bus:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance.
2. Follow the procedure in Section C.7.6.3 to make the HSZ40 inactive.
3. The HSZ40 can be powered down, but it must remain plugged-in to the power distribution system, to maintain grounding.
4. Unscrew and remove the differential tri-link from the HSZ40.
5. Protect all exposed connector pins from ESD and from contacting any electrical conductor while they are disconnected.

To insert an HSZ40 controller onto an active SCSI bus:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance. Also, ensure that the ground offset voltages between the HSZ40 and all components that will be attached to it are within the limits specified in Section C.7.8.
2. Protect all exposed connector pins from ESD and from contacting any electrical conductor while they are disconnected.
3. Power up the HSZ40 and ensure that the disk units associated with the HSZ40 conform to the following:
   - The disk units are unique for the logical SCSI bus
   - The disk units are already configured as ‘‘DK’’ devices on the following:
     - Any member of the VMScluster system that already has that ID configured
     - Any OpenVMS processor on the same SCSI bus that is running the MSCP server
4. Ensure that the HSZ40 will make a legal stubbing connection to the active segment. (The connection is legal when the tri-connector is attached directly to the HSZ40 controller module, with no intervening cable.)
5. Attach the differential tri-link to the HSZ40, using care to ensure that it is properly aligned. Tighten the screws.
6. Configure the HSZ40 virtual disks on VMScluster members, as required, using SYSMAN IO commands.

C.7.6.6 Procedure for Hot Plugging Host Adapters

To remove a host adapter from an active SCSI bus:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance.
2. Verify that the connection to be broken is a stubbing connection. If not, then hot plugging must not be performed.
3. Follow the procedure in Section C.7.6.3 to make the host adapter inactive.
4. The system can be powered down, but it must remain plugged-in to the power distribution system, to maintain grounding.
5. Remove the ''Y'' cable from the host adapter's single-ended connector.
6. Protect all exposed connector pins from ESD and from contacting any electrical conductor while they are disconnected.
7. Do *not* unplug the adapter from the host's internal bus while the host remains powered up.

   At this point, the adapter has disconnected from the SCSI bus. To remove the adapter from the host, power down the host first, and then remove the adapter from the host's internal bus.

To insert a host adapter onto an active SCSI bus:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance. Also, ensure that the ground offset voltages between the host and all components that will be attached to it are within the limits specified in Section C.7.8.
2. Protect all exposed connector pins from ESD and from contacting any electrical conductor while they are disconnected.
3. Ensure that the host adapter will make a legal stubbing connection to the active segment (the stub length must be within allowed limits, and the host adapter must not provide termination to the active segment).
4. Plug the adapter into the host (if it is unplugged).
5. Plug the system into the power distribution system, to ensure proper grounding. Power up if desired.
6. Attach the ''Y'' cable to the host adapter, using care to ensure that it is properly aligned.

C.7.6.7 Procedure for Hot Plugging DWZZAs

Use the following procedure to remove a DWZZA from an active SCSI bus:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance.
2. Verify that the connection to be broken is a stubbing connection. If not, then hot plugging must not be performed.
3. Do not power down the DWZZA. This can disrupt the operation of the attached SCSI bus segments.

4. Determine which SCSI bus segment will remain active after the disconnection. Follow the procedure in Section C.7.6.3 to make the other segment inactive.

   When the DWZZA is removed from the active segment, the inactive segment must remain inactive until the DWZZA is also removed from the inactive segment, or proper termination is restored to the DWZZA port that was disconnected from the active segment.

5. The next step depends on the type of DWZZA and the segment that is being hot plugged, as follows:

DWZZA Type	Condition	Action
DWZZA-VA	Single-ended segment will remain active	Squeeze the clips on the side of the SBB, and slide the DWZZA-VA out of the StorageWorks shelf.
DWZZA-VA	Differential segment will remain active	Unscrew and remove the differential tri-link from the DWZZA-VA.
DWZZA-AA	Single-ended segment will remain active	Remove the ''Y'' cable from the DWZZA-AA's single-ended connector.
DWZZA-AA	Differential segment will remain active	Unscrew and remove the differential tri-link from the DWZZA-AA.

6. Protect all exposed connector pins from ESD and from contacting any electrical conductor while they are disconnected.

To insert a DWZZA onto an active SCSI bus:

1. Use an ESD grounding strap that is attached either to a grounding stud or to unpainted metal on one of the cabinets in the system. Refer to the system installation procedures for guidance. Also, ensure that the ground offset voltages between the DWZZA-AA and all components that will be attached to it are within the limits specified in Section C.7.8.
2. Protect all exposed connector pins from ESD and from contacting any electrical conductor while they are disconnected.
3. Ensure that the DWZZA will make a legal stubbing connection to the active segment (the stub length must be within allowed limits, and the DWZZA must not provide termination to the active segment).
4. The DWZZA must be powered up. The SCSI segment that is being added must be attached and properly terminated. All devices on this segment must be inactive.
5. The next step depends on the type of DWZZA, and which segment is being hot plugged, as follows:

DWZZA Type	Condition	Action
DWZZA-VA	Single-ended segment is being hot plugged	Slide the DWZZA-VA into the StorageWorks shelf.
DWZZA-VA	Differential segment is being hot plugged	Attach the differential tri-link to the DWZZA-VA, using care to ensure that it is properly aligned. Tighten the screws.

DWZZA Type	Condition	Action
DWZZA-AA	Single-ended segment is being hot plugged	Attach the ''Y'' cable to the DWZZA-AA, using care to ensure that it is properly aligned.
DWZZA-AA	Differential segment is being hot plugged	Attach the differential tri-link to the DWZZA-VA, using care to ensure that it is properly aligned. Tighten the screws.

6. If the newly attached segment has storage devices on it, then configure them on VMScluster members, if required, using SYSMAN IO commands.

C.7.7 OpenVMS Requirements for Devices Used on Multiple-Host SCSI VMScluster Systems

At this time, the only devices approved for use on multiple-host SCSI VMScluster systems are those listed in Table C–2. While not specifically approved for use, other disk devices might be used in a multiple-host VMScluster system when they conform to the following requirements:

- Support for concurrent multi-initiator I/O.
- Proper management for the following states or conditions on a per-initiator basis:
  - Synchronous negotiated state and speed
  - Width negotiated state
  - Contingent Allegiance and Unit Attention conditions
- Tagged Command Queuing. This is needed to provide an ordering guarantee used in VMScluster systems to ensure that I/O has been flushed. The drive must implement queuing that complies with Section 7.8.2 of the SCSI–2 Standard, which says (in part):

  ''...All commands received with a simple queue tag message prior to a command received with an ordered queue tag message, *regardless of initiator*, shall be executed before that command with the ordered queue tag message.'' (Emphasis added.)
- Support for command disconnect.
- A reselection timeout procedure compliant with Option b of Section 6.1.4.2 of the SCSI–2 Standard. Furthermore, the device shall implement a reselection retry algorithm that limits the amount of bus-time spent attempting to reselect a non-responsive initiator.
- Automatic read reallocation enabled (ARRE) and automatic write reallocation enabled (AWRE), (that is, drive-based bad block revectoring), to prevent multiple hosts from unnecessarily revectoring the same block. To avoid data corruption, it is essential that the drive comply with Section 9.3.3.6 of the SCSI–2 Standard, which says (in part):

  ''...The automatic reallocation shall then be performed only if the target *successfully recovers the data.*'' (Emphasis added.)
- Storage devices should not supply TERMPWR. If they do, then it is necessary to apply configuration rules to ensure that there are no more than four sources of TERMPWR on a segment.

Finally, if the device or any other device on the same segment will be hot plugged, then the device must meet the electrical requirements described in Section C.7.6.2.

C.7.8 Grounding Requirements

This section describes the grounding requirements for electrical systems in a SCSI VMScluster system.

Improper grounding can result in voltage differentials, called ground offset voltages, between the enclosures in the configuration. Even small ground offset voltages across the SCSI interconnect (as shown in Step 3 in Table C–9) can disrupt the configuration, and the user may experience performance degradation or data corruption.

Table C–9 describes important considerations to ensure proper grounding.

Table C–9 Steps for Ensuring Proper Grounding

	Description
❶	Ensure that site power distribution meets all local electrical codes.
❷	Inspect the entire site power distribution system to ensure that: • All outlets have power ground connections • A grounding prong is present on all computer equipment power cables • Power outlet neutral connections are not actual ground connections • All grounds for the power outlets are connected to the same power distribution panel • All devices that are connected to the same circuit breaker as the computer equipment are UL or IEC approved
❸	If you have difficulty verifying these conditions, you can use a hand-held multimeter to measure the ground offset voltage between any two cabinets. To measure the voltage, connect the multimeter leads to unpainted metal on each enclosure. Then, determine whether the voltage exceeds the following allowable ground offset limits:

SCSI Signaling Method	Maximum Allowable Offset
Single-ended	50 millivolts
Differential	800 millivolts

	The multimeter method provides data for only the moment it is measured. The ground offset values may change over time as additional devices are activated or plugged into the same power source. To ensure that the ground offsets remain within acceptable limits over time, Digital recommends that you have a power survey performed by a qualified electrician.
❹	If you are uncertain about the grounding situation or if the measured offset exceeds the allowed limit, Digital recommends that a qualified electrician correct the problem. It may be necessary to install grounding cables between enclosures to reduce the measured offset.
❺	If an unacceptable offset voltage was measured and a ground cable was installed, then measure the voltage again to ensure it is less than the allowed limits. If not, an electrician must determine the source of the ground offset voltage and reduce or eliminate it.

D

VMScluster Systems That Span Multiple Sites—OpenVMS Version 6.2 Feature

This appendix discusses multiple-site VMScluster configurations, with an emphasis on the new wide area network ATM and DS3 communications services. It provides configuration guidelines and system management suggestions for VMScluster systems in which multiple nodes are located at sites separated by relatively long distances.

The information in this appendix supersedes the Multiple-Site VMScluster Systems addendum manual, and it supplements current multiple-site VMScluster information in the following VMScluster manuals:

- *VMScluster Systems for OpenVMS*
- *Guidelines for VMScluster Configurations*

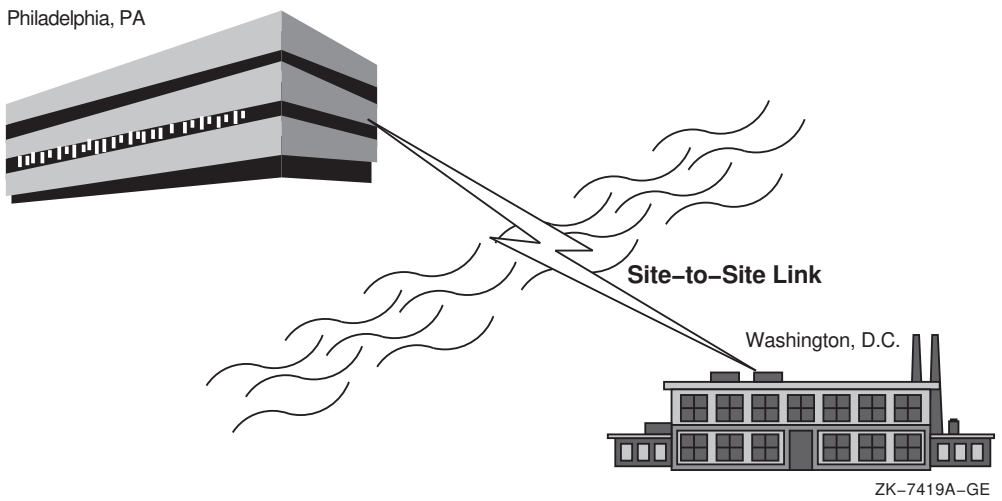
The sections that follow describe multiple-site VMScluster configurations and some of the benefits you can derive from them.

D.1 What Is a Multiple-Site VMScluster System?

A **multiple-site VMScluster system** is a VMScluster system in which the member nodes are located in geographically separate sites. When an organization has geographically disperse sites, a multiple-site VMScluster system allows the organization to realize the benefits of VMScluster systems (for example, sharing data among sites while managing data center operations at a single, centralized location).

Figure D–1 illustrates the concept of a multiple-site VMScluster system for a company with a manufacturing site located in Washington, D.C. and corporate headquarters in Philadelphia. This configuration spans a geographical distance of approximately 130 miles (210 km).

Figure D–1 Site-to-Site Link Between Philadelphia and Washington



The Fiber Distributed Data Interface (FDDI) has been in general use since VMS Version 5.4–3 to carry out cluster communications over distances of approximately 25 miles (approximately 40 km).[1]

D.1.1 ATM, DS3, and FDDI Intersite Links

The following link technologies between sites are approved for OpenVMS VAX and OpenVMS AXP systems:

- Asynchronous Transfer Mode (ATM)
- DS3
- FDDI

High-performance local area network (LAN) technology combined with the ATM, DS3, and FDDI interconnects allows you to utilize wide area network (WAN) communication services in your VMScluster configuration. VMScluster systems configured with the GIGAswitch crossbar switch and ATM, DS3, or FDDI interconnects approve the use of nodes located miles apart.[2] Section D.3 describes VMScluster systems and the WAN communications services in more detail.

Note

To gain the benefits of disaster tolerance across a multiple-site VMScluster, use the Business Recovery Server combined with Volume Shadowing for OpenVMS.

Consult your Digital Services Group or see the Software Product Descriptions (SPDs) for complete and up-to-date details about these products.

[1] The cable route distance between sites.

[2] The actual distance between any two sites is determined by the physical intersite cable-route distance, and not the straight-line distance between the sites.

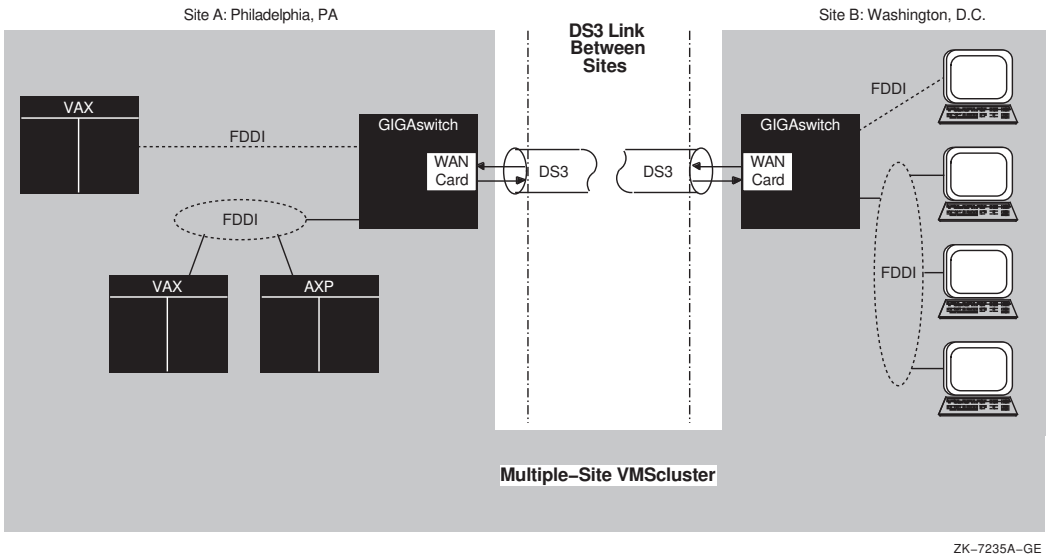
D.1.2 Benefits of Multiple-Site VMScluster Systems

Some of the benefits you can realize with a multiple-site VMScluster system include the following:

Benefit	Description
Remote satellites and nodes	A few systems can be remotely located at a secondary site and can benefit from centralized system management and other resources at the primary site, as shown in Figure D–2. For example, a main office data center could be linked to a warehouse or a small manufacturing site that could have a few local nodes with directly attached site-specific devices. Alternatively, some engineering workstations could be installed in an office park across the city from the primary business site.
Data center management consolidation	A single management team can manage nodes located in data centers at multiple sites.
Physical resource sharing	Multiple sites can readily share devices such as high-capacity computers, tape libraries, disk archives, or phototypesetters.
Remote archiving	Backups can be made to archival media at any site in the cluster. A common example would be to use disk or tape at a single site to back up the data for all sites in the multiple-site VMScluster. Backups of data from remote sites can be made transparently (that is, without any intervention required at the remote site).
Increased availability	In general, a multiple-site VMScluster provides all of the availability advantages of a LAN VMScluster. (See *VMScluster Systems for OpenVMS* for information about LANs.) Additionally, by connecting multiple, geographically separate sites, multiple-site VMScluster configurations can increase the availability of a system or elements of a system in a variety of ways: • Logical volume/data availability—Volume shadowing or redundant arrays of independent disks (RAID) can be used to create logical volumes with members at both sites. If one of the sites becomes unavailable, data can remain available at the other site. • Site failover—By adjusting the VOTES system parameter, you can select a preferred site to continue automatically if the other site fails or if communications with the other site are lost. • Disaster tolerance—When combined with the software, services, and management procedures provided by the Business Recovery Server and Volume Shadowing for OpenVMS products, you can achieve a high level of disaster tolerance. The Software Product Descriptions (SPDs) for these products provide further information.

Figure D–2 shows a VMScluster system with satellites accessible from a remote site.

Figure D–2 Multiple-Site VMScluster Configuration with Remote Satellites



D.1.3 General Configuration Guidelines

The same configuration rules that apply to VMScluster systems on a LAN also apply to a multiple-site VMScluster configuration that includes ATM, DS3, or FDDI intersite interconnect. General LAN configuration rules are stated in the following documents:

- OpenVMS Cluster Software *Software Product Description*, Version 6.2 (SPD 29.78.*xx*)
- *Guidelines for VMScluster Configurations*

Some configuration guidelines are unique to multi-site VMSclusters, and these guidelines are described in Section D.3.4.

D.2 Using FDDI to Configure Multiple-Site VMScluster Systems

Since VMS Version 5.4–3, FDDI has been the most common method to connect two distant VMScluster sites. Using high-speed FDDI fiber-optic cables, you can connect sites with an intersite cable-route distance of up to 25 miles 40 km.

You can connect sites using these FDDI methods:

- To obtain maximum performance, use a full-duplex FDDI link at 100 Mb/s both ways between GIGAswitch/FDDI bridges at each site for maximum intersite bandwidth.
- To obtain maximum availability, use a dual FDDI ring at 100 Mb/s between dual attachment stations (DAS) ports of wiring concentrators or GIGAswitch /FDDI bridges for maximum link availability.
- For maximum performance and availability, use two disjoint FDDI LANs, each with dedicated host adapters and full-duplex FDDI intersite links connected to GIGAswitch/FDDI bridges at each site.

Refer to the *GIGAswitch/FDDI ATM Linecard Reference Manual* for configuration information. Additional VMScluster configuration guidelines and system management information can be found in *Guidelines for VMScluster Configurations* and *VMScluster Systems for OpenVMS*. See the *Overview of OpenVMS Documentation* for information about ordering the current version of these manuals.

The inherent flexibility of VMScluster systems and improved VMScluster LAN protocols also allow you to connect multiple VMScluster sites using the ATM and/or DS3 communications services.

D.3 Using WAN services to Configure Multiple-Site VMScluster Systems

This section provides an overview of the ATM and DS3 wide area network (WAN) services, describes how you can bridge an FDDI interconnect to the ATM and/or DS3 communications services, and provides guidelines for using these services to configure multiple-site VMScluster systems.

The ATM and DS3 services provide long-distance, point-to-point communications that you can configure into your VMScluster system to gain WAN connectivity. The ATM and DS3 services are available from most common telephone service carriers and other sources.

Note

DS3 is not available in Europe and some other locations. Also, ATM is a new and evolving standard, and ATM services might not be available in all localities.

ATM and DS3 services are approved for use with the following OpenVMS versions:

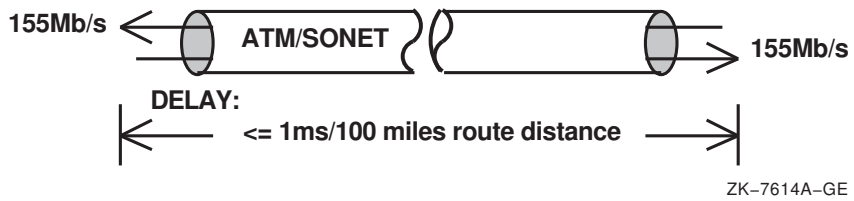
Service	Approved Versions of OpenVMS
ATM	OpenVMS Version 6.2 or later
DS3	OpenVMS Version 6.1 or later

The following sections describe the ATM and DS3 communication services and how to configure these services into multiple-site VMScluster systems.

D.3.1 The ATM Communications Service

The ATM communications service that uses the SONET physical layer (ATM /SONET) provides full-duplex communications (that is, the bit rate is available simultaneously in both directions as shown in Figure D–3). ATM/SONET is compatible with multiple standard bit rates. The SONET OC-3 service at 155 Mb/s full-duplex rate is the best match to FDDI's 100 Mb/s bit rate. ATM/SONET OC-3 is a standard service available in most parts of the world. In Europe, ATM/SONET is a high performance alternative to the older E3 standard.

Figure D–3 ATM/SONET OC-3 Service

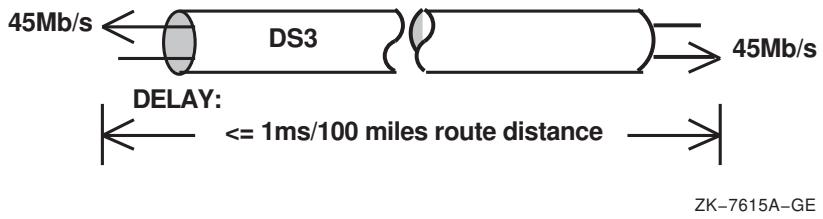


To transmit data, ATM frames (packets) are broken into **cells** for transmission by the ATM service. Each cell has 53 bytes, of which 5 bytes are reserved for header information and 48 bytes are available for data. At the destination of the transmission, the cells are reassembled into ATM frames. The use of cells permits ATM suppliers to multiplex and demultiplex multiple data streams efficiently at differing bit rates. This conversion of frames into cells and back is transparent to higher layers.

D.3.2 The DS3 Communications Service

The DS3 communications service provides full-duplex communications as shown in Figure D–4. DS3 (also known as T3) provides the T3 standard bit rate of 45 Mb/s. T3 is the standard service available in North America and many other parts of the world.

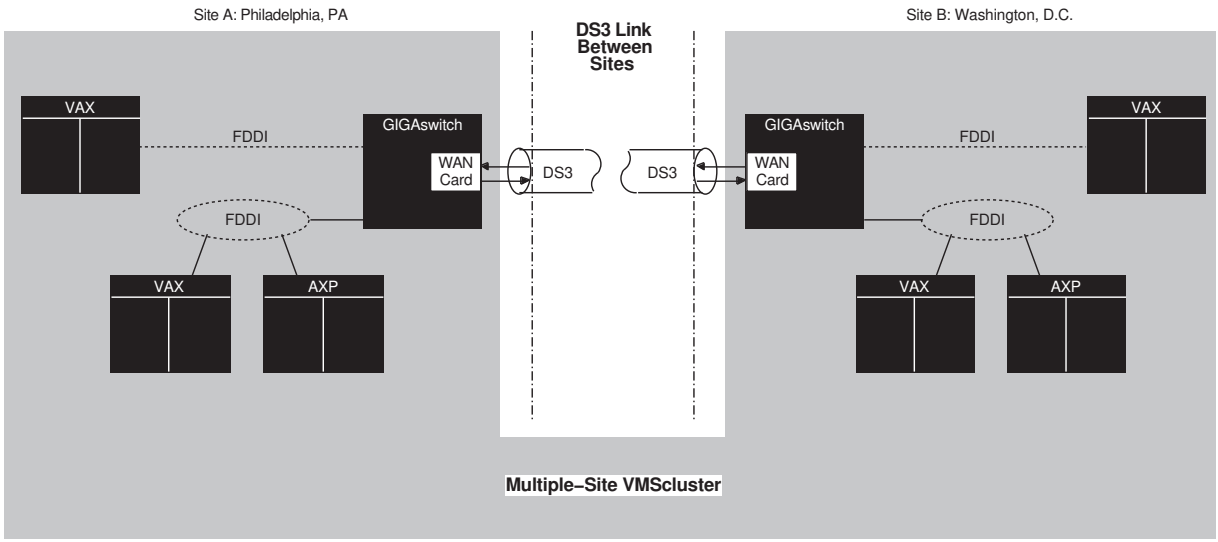
Figure D–4 DS3 Service



D.3.3 FDDI-to-WAN Bridges

You can use FDDI-to-WAN (for example, FDDI-to-ATM and/or FDDI-to-DS3) bridges to configure a VMScluster with nodes in geographically separate sites, such as the one shown in Figure D–5. In this figure, the VMScluster nodes at each site communicate as though the two sites are connected by FDDI. The FDDI-to-WAN bridges make the existence of ATM and DS3 transparent to the VMScluster software.

Figure D–5 Multiple-Site VMScluster Configuration Connected by DS3



In Figure D–5, the FDDI-to-DS3 bridges and DS3 operate as follows:

1. The local FDDI-to-DS3 bridge receives FDDI packets addressed to nodes at the other site.
2. The bridge converts the FDDI packets into DS3 packets and sends the packets to the other site via the DS3 link.
3. The receiving FDDI-to-DS3 bridge converts the DS3 packets into FDDI packets and transmits them on an FDDI ring at that site.

Digital recommends using the GIGAswitch/FDDI system to construct FDDI-to-WAN bridges. Digital used the GIGAswitch/FDDI, combined with the DEFGT WAN T3/SONET option card, during qualification testing of the ATM and DS3 communications services in multi-site VMScluster systems.

D.3.4 Guidelines for Configuring ATM and DS3 in a VMScluster System

When configuring a multiple-site VMScluster configuration, you must ensure that the intersite link's delay, bandwidth, availability, and bit error rate characteristics meet application needs. This section describes the requirements and provides recommendations for meeting those requirements.

D.3.4.1 Requirements

To be a configuration approved by Digital, a multiple-site VMScluster must comply with the following rules:

Maximum intersite link route distance	The total intersite link cable route distance between members of a multiple-site VMScluster cannot exceed 150 miles (242 km). You can obtain exact distance measurements from your ATM or DS3 supplier. *This distance restriction may be exceeded by Business Recovery Server configurations that meet Business Recovery Server configuration rules.*

Maximum intersite link utilization	Average intersite link utilization in either direction must be less than 80% of the link's bandwidth in that direction for any 10-second interval. Exceeding this utilization is likely to result in intolerable queuing delays or packet loss.
Intersite link specifications	The intersite link must meet the VMScluster requirements specified in Table D–3.
VMScluster LAN configuration rules	Apply the configuration rules for VMScluster systems on a LAN to a configuration. Documents describing configuration rules are referenced in Section D.1.3.

D.3.4.2 Recommendations

When configuring the DS3 interconnect, apply the configuration guidelines for VMScluster systems interconnected by LAN that are stated in the cluster Software Product Descriptions (SPDs) and in the *Guidelines for VMScluster Configurations* manual. VMScluster members at each site can include any mix of satellites, systems, and other interconnects such as CI and DSSI.

This section provides additional recommendations for configuring a multiple-site VMScluster system.

DS3 link capacity/protocols

The GIGAswitch with the WAN T3/SONET option card provides a full-duplex 155 Mb/s ATM/SONET link. The entire bandwidth of the link is dedicated to the WAN option card. However, The GIGAswitch/FDDI's internal design is based upon full duplex extensions to FDDI. Thus the GIGAswitch/FDDI's design limits the ATM/SONET link's capacity to 100 Mb/s in each direction.

The GIGAswitch with the WAN T3/SONET option card provides several protocol options that can be used over a DS3 link. Use the DS3 link in clear channel mode, which dedicates its entire bandwidth to the WAN option card. The DS3 link capacity varies with the protocol option selected. Protocol options are described in Table D–1.

Table D–1 DS3 Protocol Options

Protocol Option	Link Capacity
ATM[1] AAL–5[2] mode with PLCP[3] disabled.	39 Mb/s
ATM AAL–5 mode with PLCP enabled.	33 Mb/s
HDLC[4] mode (not currently available).	43 Mb/s

[1]Asynchronous Transfer Mode

[2]ATM Adaptation Layer

[3]Physical Layer Convergence Protocol

[4]High-Speed Datalink Control

For maximum link capacity, Digital recommends configuring the WAN T3/SONET option card to use ATM AAL–5 mode with PLCP disabled.

Intersite bandwidth

The intersite bandwidth can limit application locking and I/O performance (including volume shadowing or RAID set copy times) and the performance of the lock manager.

To promote reasonable response time, Digital recommends that average traffic in either direction over an intersite link not exceed 60% of the link's bandwidth in that direction for any 10-second interval. Otherwise, queuing delays within the FDDI-to-WAN bridges can adversely affect application performance.

Remember to account for both VMScluster communications (such as locking and I/O) and network communications (such as TCP/IP, LAT, and DECnet) when calculating link utilization.

Intersite delay

An intersite link introduces a one-way delay of up to 1 ms per 100 miles of intersite cable route distance plus the delays through the FDDI-to-WAN bridges at each end. Digital recommends that you consider the effects of intersite delays on application response time and throughput.

For example, intersite link one-way path delays have the following components:

- Cable route one-way delays of 1 ms/100 miles (0.01 ms/mile) for both ATM and DS3.
- FDDI-to-WAN bridge delays (approximately 0.5 ms per bridge, and 2 bridges per one-way trip)

Calculate the delays for a round trip as follows:

$$WAN\ ROUND\ TRIP\ DELAY =$$

$$2 \times (\ N\ miles\ \times\ 0.01\ ms\ per\ mile\ +\ 2\ \times\ 0.5\ ms\ per\ FDDI-WAN\ bridge\ )$$

An I/O write operation that is MSCP served requires a minimum of two round-trip packet exchanges:

$$WAN\ I/O\ Write\ Delay = 2 \times WAN\ Round\ Trip\ Delay$$

Thus, an I/O write over a 100-mile WAN link takes at least 8 ms longer than the same I/O write over a short, local FDDI.

Similarly, a lock operation typically requires a round trip exchange of packets:

$$WAN\ Lock\ Operation\ Delay = WAN\ Round\ Trip\ Delay$$

An I/O operation with *N* locks to synchronize it incurs the following delay due to WAN:

$$WAN\ Locked\ IO\ Operation\ Delay = (N\ \times\ WAN\ Lock\ Operation\ Delay) + WAN\ I/O\ Delay$$

Bit error ratio

The bit error ratio (BER) parameter is an important measure of the frequency that bit errors are likely to occur on the intersite link. You should consider the effects of bit errors on application throughput and responsiveness when configuring a multiple-site VMScluster. Intersite link bit errors can result in packets being lost and retransmitted with consequent delays in application I/O response time (see Section D.3.6). You can expect application delays ranging from a few hundred milliseconds to a few seconds each time a bit error causes a packet to be lost.

Intersite link availability

Interruptions of intersite link service can result in the resources at one or more sites becoming unavailable until connectivity is restored (see Section D.3.5).

System disks

Sites with nodes contributing quorum votes should have a local system disk or disks for those nodes.

System management

A large, multiple-site VMScluster requires a system management staff trained to support an environment that consists of a large number of diverse systems that are used by many people performing varied tasks.

Microwave DS3 links

You can provide portions of a DS3 link with microwave radio equipment. The specifications in Section D.3.6 apply to any DS3 link. The BER and availability of microwave radio portions of a DS3 link are affected by local weather and the length of the microwave portion of the link. Consider working with a microwave consultant who is familiar with your local environment if you plan to use microwaves as portions of a DS3 link.

D.3.5 Availability Considerations

If the FDDI-to-WAN bridges and the link that connects multiple sites become temporarily unavailable, the following events could occur:

- Intersite link failures can result in the resources at one or more sites becoming unavailable until intersite connectivity is restored.
- Intersite link bit errors (and ATM cell losses) and unavailability can affect:
  - System responsiveness
  - System throughput (or bandwidth)
  - Virtual circuit (VC) closure rate
  - VMScluster transition and site failover time

Many communication service carriers offer availability-enhancing options, such as path diversity, protective switching, and other options that can significantly increase the intersite link's availability.

D.3.6 Specifications

This section describes the requirements for successful communications and performance with the WAN communications services.

To assist you in communicating your requirements to a WAN service supplier, this section uses WAN specification terminology and definitions commonly used by telecommunications service providers. These requirements and goals are derived from a combination of Bellcore Communications Research specifications and a Digital analysis of error effects on VMSclusters.

Table D–2 describes terminology that will help you understand the Bellcore and VMScluster requirements and goals used in Table D–3.

Use the Bellcore and VMScluster requirements for ATM/SONET - OC3 and DS3 service error performance (quality) specified in Table D–3 to help you assess the impact of the service supplier's service quality, availability, down time, and service-interruption frequency goals on the system.

Note

To ensure that the VMScluster system meets your application response-time requirements, you might need to establish WAN requirements that exceed the Bellcore and VMScluster requirements and goals stated in Table D–3.

Table D–2 Bellcore and VMScluster Requirements and Goals Terminology

Specification	Requirements	Goals
Bellcore Communications Research	Bellcore specifications are the recommended ''generic error performance requirements and objectives'' documented in the Bellcore Technical Reference TR–TSY–000499 *TSGR: Common Requirements*. These specifications are adopted by WAN suppliers as their service guarantees. The FCC has also adopted them for tariffed services between common carriers. However, some suppliers will contract to provide higher service-quality guarantees at customer request. Other countries have equivalents to the Bellcore specifications and parameters.	These are the recommended minimum values. Bellcore calls these goals their ''objectives'' in the *TSGR: Common Requirements* document.
VMScluster	In order for Digital to approve a configuration, parameters must meet or exceed the values shown in the VMScluster Requirements column in Table D–3.	For optimal VMScluster operation, all parameters should meet or exceed the VMScluster Goal values. Note that if these values are met or exceeded, then interconnect bit errors and bit error recovery delays should not significantly degrade average VMScluster throughput. VMScluster response time should be generally satisfactory, although there may be brief application pauses a few times **per day**.[2]

IF...	THEN...
These values are not met	VMScluster performance will probably be unsatisfactory because of interconnect errors/error recovery delays, and VC closures that may produce VMScluster state transitions and/or site failover.
These values are met or exceeded	Interconnect bit error–related recovery delays will not significantly degrade average VMScluster throughput. VMScluster response time should be generally satisfactory. Note that if the requirements are only being met, there may be several application pauses **per hour**. [1]

[1]Pauses are due to a virtual circuit retransmit timeout resulting from a lost packet on one or more NISCA transport virtual circuits. Each pause might last from a few hundred milliseconds to a few seconds.

[2]Application pauses may occur every hour or so (similar to what is described under VMScluster Requirements) because of packet loss caused by bit error.

Table D–3 VMScluster DS3 & SONET OC3 Error Performance Requirements

Parameter	Bellcore Requirement	Bellcore Goal	VMScluster Requirement[1]	VMScluster Goal[1]	Units
Errored seconds (% ES)	<1.0%	<0.4%	<1.0%	<0.028%	% ES/24 hr
	The ES parameter can also be expressed as a count of errored seconds, as follows:				
	<864	<345	<864	<24	ES per 24-hr period
Burst errored seconds (BES)[2]	$\leq$4	–	$\leq$4	Bellcore Goal	BES/day
Bit error ratio (BER)[3]	1×10^{-9}	2×10^{-10}	1×10^{-9}	6×10^{-12}	Errored bits/bit
DS3 Channel unavailability	None	$\leq$97 @ 250 miles, linearly decreasing to 24 @ $\leq$25 miles	None	Bellcore Goal	Min/yr
SONET Channel unavailability	None	$\leq$105 @ 250 miles, linearly decreasing to 21 @ $\leq$50 miles	None	Bellcore Goal	Min/yr
Channel unavailable event[4]	None	None	None	1 to 2	Events/year

[1]Application requirements might need to be more rigorous than those shown in the VMScluster Requirements column.

[2]Averaged over many days.

[3]Does not include any burst errored seconds occurring in the measurement period.

[4]The average number of channel downtime periods occurring during a year. This parameter is useful for specifying how often a channel might become unavailable.

Table Key

- Availability—The long-term fraction or percentage of time that a transmission channel performs as intended. Availability is frequently expressed in terms of unavailability or down time.
- BER (bit error ratio)—''The BER is the ratio of the number of bits in error to the total number of bits transmitted during a measurement period, excluding all burst errored seconds (defined below) in the measurement period. During a burst errored second, neither the number of bit errors nor the number of bits is counted.''
- BES (burst errored second)—''A burst errored second is any errored second containing at least 100 errors.''
- Channel—The term for a link that is used in the Bellcore *TSGR: Common Requirements* document for a SONET or DS3 link.
- Down time—The long-term average amount of time (for example, minutes) that a transmission channel is not available during a specified period of time (for example, 1 year).

  ''...unavailability or downtime of a channel begins when the first of 10 [or more] consecutive Severely Errored Seconds (SESs) occurs, and ends when the first of 10 consecutive non-SESs occurs.''

  The unavailable time is counted from the first SES in the 10–SES sequence.

  ''The time for the end of unavailable time is counted from the first fault-free second in the [non-SES] sequence.''
- ES (errored second)—''An errored second is any one-second interval containing at least one error.''
- SES (severely errored second)—''...an SES is a second in which the BER is greater than 10^{-3}.''

D.4 Managing VMScluster Systems Across Multiple Sites

In general, you manage a multiple-site VMScluster using the same tools and techniques that you would use for any VMScluster interconnected by a LAN. The following sections describe some additional considerations and recommends some system management tools and techniques.

The following table lists system management considerations specific to multiple-site VMScluster systems.

Problem	Possible Solution
Multiple-site configurations present an increased probability of the following failure modes: • VMScluster quorum loss resulting from site-to-site communication link failure. • Site loss resulting from power failure or other breakdown can affect all systems at that site.	Assign votes so that one preferred site has sufficient votes to maintain quorum and to continue operation if the site-to-site communication link fails or if the other site is unavailable. Select the site with the most critical applications as the primary site. Sites with a few noncritical systems or satellites probably should not have sufficient votes to continue.
Users expect that the local resources will either continue to be available or will rapidly become available after such a failure. This might not always be the case.	Consider some of the following options for setting user expectations: • Set management and user expectations regarding the likely effects of failures, and consider training remote users in the procedures to be followed at a remote site when the system becomes unresponsive because of quorum loss or other problems. • Develop management policies and procedures for what actions will be taken to identify and handle these failure modes. These procedures may include manually adjusting quorum to allow a site to continue.

D.4.1 Methods and Tools

You can use the following system management methods and tools to manage both remote and local nodes:

- There are two options for remote-site console access when you use an intersite link through a DECserver in reverse LAT mode.
  - Use the following tools to connect remote consoles:
    - SET HOST/LAT command
    - POLYCENTER Console Manager
    - VMScluster Console System (VCS)
    - Business Recovery Server Operations Management Station (includes VCS)
  - Use a modem to dial up the remote system consoles.
- An alternative to remote-site console access is to have a system manager at each site.
- To enable device and processor control commands to take effect across all nodes in a VMScluster system, use the System Management utility (SYSMAN) that is supplied with the OpenVMS operating system.

D.4.2 Shadowing Data

Volume Shadowing for OpenVMS allows you to shadow data volumes across multiple sites. System disks can be members of a volume shadowing or RAID set within a site; however, use caution when configuring system disk shadow set members in multiple sites. This is because it may be necessary to boot off a remote system disk shadow set member after a failure. If your system does not support FDDI booting, it will not be possible to do this.

See the Software Product Descriptions (SPDs) for complete and up-to-date details about Volume Shadowing for OpenVMS and StorageWorks RAID for OpenVMS.

D.4.3 Monitoring Performance

Monitor performance for multiple-site VMScluster systems as follows:

- Monitor the virtual circuit (VC) packet-loss count and round-trip time values using the System Dump Analyzer (SDA). The procedures for doing this are documented in *VMScluster Systems for OpenVMS*.
- Monitor the intersite link bit error ratio (BER) and packet loss using network management tools. You can use tools such as POLYCENTER NetView or DECmcc to access the GIGAswitch and WAN T3/SONET option card's management information and to set alarm thresholds. See the GIGAswitch, WAN T3/SONET card, POLYCENTER, and DECmcc documentation, as appropriate.

E

Other OpenVMS Version 6.2 New Features

This appendix contains descriptions of OpenVMS Version 6.2 new features not otherwise documented in the printed manuals for OpenVMS Version 6.2 or 7.0.

E.1 VMScluster Systems New Features

In OpenVMS Version 6.2, VMScluster systems included the following new features:

- OpenVMS Cluster Client Software
- Support for TMSCP served SCSI tapes
- Enhanced support for HSJ, HSC, and HSD series controller failover

These features further enhanced the performance, availability, and functionality of VMScluster systems. The following sections describe these features in more detail.

E.1.1 OpenVMS Cluster Client Software

OpenVMS Version 6.2 introduces a new license type for OpenVMS cluster software, called OpenVMS Cluster Client software. The new license provides a low-cost cluster client product for Alpha and VAX workstations.

The OpenVMS Cluster Client software provides fully functional OpenVMS Cluster software with two exceptions:

- Clients cannot provide VOTES to the VMScluster configurations
- Clients cannot MSCP serve disks or TMSCP serve tapes

OpenVMS Cluster Client software is available for VAX and Alpha systems. The software uses a License Management Facility (LMF) license name of VMSCLUSTER-CLIENT. OpenVMS Cluster Client software is included in the NAS 150 package. Refer to the OpenVMS Cluster *Software Product Description* for ordering information.

E.1.2 Support for TMSCP Served SCSI Tapes

VMScluster Systems for OpenVMS has been enhanced to allow the TMSCP server to serve SCSI tapes. The TMSCP server makes locally connected tapes of the following types available across a cluster:

- TA series tapes for CI
- TF series tapes for DSSI
- TZ and TLZ tapes for SCSI

The TMSCP server is controlled by the TMSCP_LOAD and TMSCP_SERVE_ALL system parameters.

- The TMSCP_LOAD parameter controls whether the TMSCP server is loaded. By default, the value of the TMSCP_LOAD parameter is set to zero so that the TMSCP server is not loaded. Refer to *VMScluster Systems for OpenVMS* for information about setting this parameter.
- The TMSCP_SERVE_ALL system parameter is new with this release. This parameter specifies TMSCP tape-serving functions when the TMSCP server is loaded. If TMSCP_LOAD is set to zero, the TMSCP_SERVE_ ALL parameter is ignored. Table E–1 describes the TMSCP_SERVE_ALL parameter settings.

Table E–1 TMSCP_SERVE_ALL System Parameter Settings

Value	Description
0	Serve no tapes. This is the default value.
1	Serve all available tapes.
2	Serve only locally connected tapes.

E.1.2.1 No TMSCP Server Support for SCSI Retension Command

The SCSI retension command modifier is not supported by the TMSCP server. Retension operations should be performed from the node serving the tape.

E.1.3 Enhanced Support for HSJ, HSC, and HSD Series Controller Failover

In previous releases of VMScluster software, dual porting of disks between pairs of HSJ and HSC series controllers was supported when the controllers were attached to a common star coupler. Beginning with OpenVMS Version 6.2, you can connect dual-ported disks to pairs of HSJ and HSC series controllers that are attached to different star couplers. This feature enhances availability because failure of a CI adapter need not cause both controllers to become unreachable.

This feature is available for HSJ and HSC series CI controllers and for HSD30 DSSI controllers. It permits HSD30 controllers to be configured across different DSSI buses.

Index

A

Adapters
  add on SCSI, C–9
  integral SCSI, C–18
Affinity
  explicit, 4–27
  implicit, 4–28
Allocation classes
  setting for SCSI configurations, C–4, C–20, C–22
ALPHA$LIBRARY logical name, 4–11
ALPHA$LOADABLE_IMAGES logical name, 4–11
/ALPHA qualifier, 4–12
Arbitration rules, for control of SCSI bus, C–36
  modifying the effect of, C–36
Architectures
  linker options, 4–11
ATM communications service, D–5
  system management, D–12
ATM Intersite Link Specifications, D–10

B

64-Bit System Services, 4–1
64-Bit Virtual Addressing, 4–1
Buffers
  using in a LAT environment, 3–9

C

Capabilities
  system, 4–24
  user, 4–25
Character set description (charmap) file
  components, B–45
  descriptions of, B–42 to B–47
Communications services
  ATM, D–5
  DS3, D–5
Configurations
  building SCSI VMScluster systems with DWZZA converters, C–11
  building with add on SCSI adapters, C–9
  building with an HSZ40 controller, C–12
Configurations (cont'd)
  building with BA350/BA353 StorageWorks enclosures, C–9
  building with internal SCSI adapters, C–18
  installing SCSI VMScluster systems, C–19
  multiple-host SCSI access on VMScluster system, C–3
  SCSI concepts, C–6
  SCSI hardware, C–9
  SCSI interconnect requirements, C–4
  SCSI VMScluster systems, C–1
  single-ended and differential SCSI signaling, C–7
  single-host SCSI access on VMScluster system, C–2
  troubleshooting SCSI VMScluster systems, C–33
  unique SCSI device IDs, C–6
  using CLUSTER_CONFIG for SCSI VMScluster systems, C–26
  WANs, D–2
Controllers
  HSZ40 controllers, C–12
Converters
  using DWZZA in SCSI VMScluster systems, C–11
COPY command, 4–22
CPU scheduling, 4–24
$CPU_CABILITIES service, 4–26
CREATE command, 4–21
CREATE/DIRECTORY command
  /ALLOCATION qualifier, 2–1
Creating large directories, 2–1
$CREPRC system service
  new node argument, 4–23
Cross-architecture
  linking, 4–11, 4–12
    logical names, 4–11
CWCREPRC_ENABLE system parameter, 3–1

D

DBGTK_SCRATCH system parameter, 3–1
DCL commands
  RUN [process] command
    /ON qualifier, 2–1

Debugger
CALL command and floating-point parameters, 4–4
Close File menu item, 4–6
customization features, 4–4
customizing fonts, 4–6
Editor File menu, 4–6
EXAMINE/DEFINITIONS command, 4–2
internationalization, 4–3
customizing fonts, 4–6
null frame procedures, 4–4
Refresh File menu item, 4–6
SHOW CALLS command, 4–4
SHOW STACK command, 4–4
with optimized programs, 4–1
semantic events, 4–2
SET STEP SEMANTIC_EVENT command, 4–3
split-lifetime variables, 4–2
STEP/SEMANTIC_EVENT command, 4–3
Debugger, 4–6
DECamds
New Cluster Windows, 6–5
Single Disk Summary Window, 6–3
summary of features, 6–1
System Overview Window, 6–2
DECnet, 3–12
DECnet/OSI, 3–12
DECthreads, 4–11
POSIX 1003.1c standard, 4–7
Thread Independent Services (TIS), 4–7
Delta/XDelta Debugger (DELTA/XDELTA)
debugging multithreaded applications, 4–8
Device IDs
configuring for SCSI, C–21
Differential signaling
for SCSI, C–7
Digital TCP/IP, 3–12
Directory blocks
controlling read/write checking of, 3–2
Disks
accessing SCSI, C–2
SCSI, C–1
SCSI concepts, C–6
SCSI configuration requirements, C–4
Display changes, 3–28
DS3 communications service, D–5
system management, D–12
DS3 Intersite Link Specifications, D–10
Dump file compression, 4–18
DUMPSTYLE parameter, 4–19
DUMP subset, 4–19

E

EFN$C_ENF event flag, 4–30
EFN 128, 4–30
Error-handling requirements, D–10
ERRORLOGBUFFERS system parameter, 4–21
Error messages, A–1
Event flag 128, 4–30

F

Fast IO system services, 4–22
Fast Path, 5–7
FAST_PATH system parameter, 3–2
FDDI
multiple-site VMSclusters, D–4
File compression
of dump files, 4–18
File systems
Spiralog, 4–18

G

GENCAT command, B–2
$GETJPI service, 4–30
$GETSYI service, 4–29
Grounding
SCSI requirements, C–20
troubleshooting, C–36

H

Hardware
add on SCSI adapters, C–9
BA350/BA353 StorageWorks enclosures, C–9
DWZZA converters, C–11
HSZ40 controllers, C–12
in SCSI VMScluster systems, C–9
integral SCSI adapters, C–18
Heap Analyzer, 4–6
Host-based RAID
support for, C–4
Host-based shadowing
support for, C–4
Host IDs
configuring for SCSI, C–20
Hosts
in a VMScluster system, C–1
Hot plugging (SCSI devices), C–38
HSC series controller failover, E–2
HSD series controller failover, E–2
HSJ series controller failover, E–2

I

ICONV commands
  COMPILE, B–6
  CONVERT, B–10
Images
  building for Alpha and VAX architectures, 4–12
  specifying VAX in link operations, 4–13
Installation
  configuring SCSI node IDs, C–20
  SCSI VMScluster systems, C–19
Interconnects
  accessing SCSI storage over, C–2
  ATM, D–5
  DS3, D–5
  FDDI, D–4
  SCSI, C–1
  troubleshooting SCSI, C–33
Internationalization of debugger, 4–3
Internet
  accessing, 2–9
IO_PREFER_CPUS system parameter, 3–1
Item codes, 4–15, 4–29, 4–30
  user context, 4–15

K

Kernel Threads, 4–10

L

LANs
  multiple-site VMScluster, D–3
  using multiple LAN adapters for LAT node, 3–5
LAT Control Program (LATCP) utility
  /[NO]ANNOUNCEMENTS qualifier, 3–11
  /[NO]LARGE_BUFFER qualifier, 3–10
LAT item codes, 4–14
LAT software
  disabling service announcements, 3–11
  rating algorithm, 3–11
  SET HOST/LAT command
    /FRAME qualifier, 3–12
  terminal speed, 3–5
  using large buffers, 3–9
  using multiple LAN adapters, 3–5
LC_COLLATE locale category, B–23
LC_CTYPE locale category, B–27
LC_MESSAGES locale category, B–30
LC_MONETARY locale category, B–31
LC_NUMERIC locale category, B–35
LC_TIME locale category, B–36
Licenses
  VMScluster client software, E–1
LOCALE commands
  COMPILE, B–11
  LOAD, B–14
  SHOW CHARACTER_DEFINITIONS, B–15
  SHOW CURRENT, B–16
  SHOW PUBLIC, B–18
  SHOW VALUE, B–18
  UNLOAD, B–15
Locale file format
  descriptions of, B–22 to B–42
  locale categories, B–22, B–23

M

MAIL$SEND_BEGIN routine, 4–15
MAIL$USER_BEGIN routine, 4–15
MAIL$USER_GET_INFO routine, 4–15
MAIL$USER_SET_INFO routine, 4–16
MAIL$_SEND_NO_SIGFILE item code, 4–15
MAIL$_SEND_SIGFILE item code, 4–15
MAIL$_USER_SET_NO_SIGFILE item code, 4–15
MAIL$_USER_SET_SIGFILE item code, 4–15
MAIL$_USER_SIGFILE item code, 4–15
MAIL command
  using /SIGNATURE_FILE qualifier, 2–7
Mail utility
  SEARCH command, 2–9
  using /PAGE qualifier, 2–8
  using signature files, 2–6
Managing multiple sites, D–13
MAXBOBMEM system parameter, 3–2
MERGE command
  high-performance Sort/Merge utility, 2–4
Messages, A–1
Multiple execution contexts, 4–10
Multithreaded applications
  debugging with DELTA/XDELTA, 4–8
MULTITHREAD system parameter, 3–2

N

/NEXT qualifier
  with SEARCH command (Mail), 2–9
Nodes
  in a VMScluster system, C–1
/NO_INITIAL_FF qualifier, 3–30

O

OPCOM (Operator Communication Manager)
  OPC$ALLOW_INBOUND logical name, 3–12
  OPC$ALLOW_OUTBOUND logical name, 3–12
OpenVMS cluster client software, E–1
Optimization and debugging, 4–1
Other command changes, 3–29

P

/PAGE=SAVE qualifier navigation keys, 2–8
/PAGE qualifier
  using with Mail utility, 2–8
Parameters
  setting SCSI, C–24
Performance
  data transfer rates, C–7
  SCSI storage, C–6
Performance requirements, D–10
Portable Character Set, B–42
Print queues, 3–30
1024 Process Identifiers, 3–30
$PROCESS_AFFINITY service, 4–28
$PROCESS_CAPABILITIES service, 4–26

R

RAID (redundant arrays of independent disks), C–2
Read and write checking
  of directory blocks, 3–2
Redundant arrays of independent disks
  See RAID
Requirements
  Bellcore Communications Research, D–11
  VMScluster, D–11
Run-time library (RTL) routines
  LIB$CREATE_DIR, 4–31

S

SAVEDUMP system parameter, 4–21
Schedulers
  callback, 4–11
  DECthreads, 4–11
  OpenVMS, 4–11
SCSI bus
  arbitration rules, C–36
  control of, C–36
SCSI disks
  accessing, C–2
  configurations requirements, C–4
  modes of operation, C–6
  unique device IDs, C–6
SCSI interconnect, C–1
  ANSI standard, C–1
  cabling and termination, C–8
  concepts, C–6
  configuration requirements, C–4
  configurations using add on SCSI adapters, C–9
  configurations using an HSZ40 controller, C–12
  configurations using BA350/BA353 StorageWorks enclosures, C–9
  configurations using DWZZA converters, C–11
SCSI interconnect (cont'd)
  configurations using integral SCSI adapters, C–18
  configuring device IDs, C–21
  configuring SCSI node IDs, C–20
  configuring with CLUSTER_CONFIG.COM, C–26
  connected to a single host, C–2
  connected to multiple hosts, C–3
  data transfer rates, C–7
  grounding requirements, C–20
  hardware configurations, C–9
  hot plugging devices with, C–38
  installation, C–19
  maximum distances, C–8
  maximum length, C–7
  number of devices supported, C–6
  performance, C–6
  power up and verify, C–22
  show and set console parameters, C–24
  TERMPWR line, C–8
  troubleshooting, C–33
SCSI tapes
  no support for SCSI retension, E–2
  support in a VMScluster system, E–1
SEARCH command
  /NEXT qualifier, 2–9
SET FETCH
  SDA command, 3–23
SET HOST/LAT command
  /FRAME qualifier, 3–12
$SET_IMPLICIT_AFFINITY service, 4–29
Shadowing
  support for, C–4
SHOW CLUSTER
  SDA command, 3–24
SHOW CONNECTIONS
  SDA command, 3–24
SHOW CPU command, 4–29
SHOW LAN
  SDA command, 3–24
SHOW LOCK
  SDA command, 3–24
SHOW MEMORY command, 4–21
SHOW PAGE_TABLE
  SDA command, 3–25
SHOW PFN_DATA
  SDA command, 3–25
SHOW PROCESS
  SDA command, 3–27
SHOW PROCESS command, 4–29
SHOW RESOURCE
  SDA command, 3–27
SHOW SUMMARY
  SDA command, 3–27
Signature file, 4–14

Signature file item codes, 4–14
Signature files
  using with Mail utility, 2–6
Single-ended signaling
  for SCSI, C–7
Site-specific startup command procedure
  releasing page file blocks, 4–22
Small Computer Systems Interface
  See SCSI
SMB$READ_MESSAGE_ITEM routine, 4–22
SMBMSG$K_PRINT_CONTROL message item code, 4–22
SMBMSG$V_NO_INITIAL_FF symbol, 4–22
SONET OC-3 Intersite Link Specifications, D–10
SOR$ routines
  high-performance Sort/Merge utility, 4–8
SORT command
  high-performance Sort/Merge utility, 2–4
Sort/Merge utility (SORT/MERGE)
  high-performance Sort/Merge utility, 2–4, 4–8
Specifications
  ATM Intersite Link, D–10
  DS3 Intersite Link, D–10
  T3 Intersite Link, D–10
  WAN Intersite Link, D–10
Spiralog file system, 4–18
SPT (system page table)
  in system dump file, 4–20
StorageWorks
  BA350/BA353 enclosures, C–9
Symbiont/Job Controller Routines, 4–22
SYS$LIBRARY logical name, 4–11
SYS$MANAGER:SYLOGICALS.COM command procedure
  OPCOM logical names, 3–12
SYS$SYSTEM:PAGEFILE.SYS file, 4–21
SYS$SYSTEM:SYSDUMP.DMP file, 4–20
System capabilities, 4–24
System Dump Analyzer (SDA)
  new and changed commands, 3–23
System dump files, 4–18 to 4–22
System management
  ATM multisite, D–12
  DS3, D–12
  methods and tools, D–13
  multisite, D–12
  WAN multisite, D–12
System messages, A–1
System page file
  as dump file, 4–21
System parameters
  ACP_DATACHECK, 3–2
  new, 3–1
  SYSTEM_CHECK, 3–2
System services
  SYS$CPU_CAPABILITIES, 4–23
  SYS$CREPRC node argument, 4–23
System services (cont'd)
  SYS$PROCESS_AFFINITY, 4–23
  SYS$PROCESS_CAPABILITIES, 4–23
  SYS$SET_IMPLICIT_AFFINITY, 4–23

T

T3 Intersite Link Specifications, D–10
Tapes
  no support for SCSI retension, E–2
  SCSI, E–1
  TMSCP served, E–2
TDF (time differential factor)
  map for determining, 3–15
Terminators
  for SCSI, C–8
Thread manager, 4–11
TMSCP servers
  controlled by TMSCP_LOAD, E–2
  controlled by TMSCP_SERVE_ALL, E–2
  SCSI tapes in a VMScluster system, E–1
TMSCP_LOAD system parameter, E–2
TMSCP_SERVE_ALL system parameter, E–2
Troubleshooting
  SCSI configurations, C–33

U

Upcall, 4–11
User capabilities, 4–25
User profile entry
  Signature file, 4–14

V

VAX
  images
    specifying in link operations, 4–13
VAX$LIBRARY logical name, 4–11
/VAX qualifier, 4–12, 4–13
VMScluster systems
  ATM communications service, D–5
  ATM multisite system management, D–12
  client software licensing, E–1
  CLUSTER_CONFIG.COM, C–26
  DS3 communications service, D–5
  DS3 system management, D–12
  hosts, nodes, and computers, C–1
  HSC series controller failover, E–2
  HSD series controller failover, E–2
  HSJ series controller failover, E–2
  installing SCSI, C–19
  mixed-architecture support, 3–30
  mixed-version support, 3–30
  SCSI configuration requirements, C–4
  SCSI hardware configurations, C–9
  SCSI performance, C–6
  SCSI storage connected to a single-host, C–2

VMScluster systems (cont'd)
SCSI storage connected to multiple-hosts, C–3
SCSI storage interconnect, C–1
serving SCSI tapes, E–1
shared SCSI storage concepts, C–6
using FDDI, D–4
WAN multisite system management, D–12
warranted and migration support, 3–30

W

WAN communications service
system management, D–12
WAN Intersite Link Specifications, D–10
WANs
utilizing in a VMScluster, D–2
Wide area networks
See WANs
Wind/U Version 3.0, 4–31

X

XPG4 localization utilities, B–1
character map (charmap) file, B–42 to B–47
GENCAT command, B–2
ICONV commands
COMPILE, B–6
CONVERT, B–10
LOCALE commands
COMPILE, B–11
LOAD, B–14
SHOW CHARACTER_DEFINITIONS, B–15
SHOW CURRENT, B–16
SHOW PUBLIC, B–18
SHOW VALUE, B–18
UNLOAD, B–15
locale file format, B–22 to B–42
ZIC command, 4–31

Z

ZIC command, 4–31
ZIC Link Lines, 4–35
ZIC parameters, 4–32
ZIC Rule Lines, 4–32 to 4–34
ZIC Zone Lines, 4–34 to 4–35