OpenVMS Software Overview

December 1995

This manual summarizes the software capabilities of the OpenVMS operating system, and describes the computing environments in which the system operates.

Revision/Update Information:	This manual supersedes the *OpenVMS Software Overview*, OpenVMS AXP Version 6.1, OpenVMS VAX Version 6.1
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: ACCESSWORKS, ACMS, ALL–IN–1, AlphaServer, AlphaStation, AXP, Bookreader, Business Recovery Server, CDA, CI, DATATRIEVE, DEC, DEC Ada, DEC BASIC, DEC C, DEC C++, DEC/EDI, DEC Fortran, DEC Notes, DEC OPS5, DEC Pascal, DEC Reliable Transaction Router, DEC RTR, DECADMIRE, DECamds, DECdns, DECdtm, DECevent, DECforms, DEChub, DEClinks, DECmessageQ, DECmigrate, DECnet, DECNIS, DEComni, DECplan, DECquery, DECram, DECserver, DECset, DECtalk, DECterm, DECthreads, DECtp, DECwindows, Digital, Digital GKS, Digital Open3D, Digital PHIGS, Digital VTX, DNA, EDT, eXcursion, GIGAswitch, HSC, InfoServer, LAT, LinkWorks, MAILbus, MailWorks, MicroVAX, MSCP, ObjectBroker, OpenVMS, PATHWORKS, POLYCENTER, PrintServer, Reliable Transaction Router, rtVAX, StorageWorks, TMSCP, TURBOchannel, ULTRIX, VAX, VAX APL, VAX BASIC, VAX C, VAX DIBOL, VAX DOCUMENT, VAX MACRO, VAXcluster, VAXELN, VAXsimPLUS, VAXstation, VMS, VMScluster, VT, WPS, WPS–PLUS, XUI, and the DIGITAL logo.

The following are third-party trademarks: AIX, DB2, IBM, OS/2, and PROFS are registered trademarks and IMS is a trademark of International Business Machines Corporation. AppleShare, AppleTalk, LocalTalk, and Macintosh are registered trademarks of Apple Computer, Inc. BASIC is a registered trademark of the Trustees of Dartmouth College, D.B.A. Dartmouth College. BSD is a trademark of the University of California, Berkeley, CA. Display PostScript and PostScript are registered trademarks of Adobe Systems Incorporated. HP and HP–UX are registered trademarks of Hewlett-Packard Company. IEEE and POSIX are registered trademarks of the Institute of Electrical and Electronics Engineers. IPX is a registered trademark of Ideographics, Inc. MCI Mail is a registered trademark of MCI Communications Corporation. Microsoft, MS, MS–DOS, MS–Windows, and Windows 95 are registered trademarks, and Windows and Windows NT are trademarks of Microsoft Corporation. MIT is a registered trademark and X Window System is a trademark of the Massachusetts Institute of Technology. Mosaic is a trademark of the University of Illinois. Motif, OSF, OSF/1, and OSF/Motif are registered trademarks of the Open Software Foundation, Inc. MUMPS is a registered trademark of Massachusetts General Hospital. NetBIOS is a trademark of Microcomputer Systems, Inc. NetWare and Novell are registered trademarks of Novell, Inc. NFS and Sun are registered trademarks, and SPARCstation and SunOS are trademarks of Sun Microsystems, Inc. Oracle is a registered trademark, and Oracle CODASYL DBMS and Oracle Rdb are trademarks of Oracle Corporation. OSI is a registered trademark of CA Management, Inc. SCO is a trademark of Santa Cruz Operations, Inc. Sybase is a registered trademark of Sybase, Inc. Telnet is a registered trademark of American Telnet, Inc. TYMNET is a registered trademark of British Telecom. UNIX is a registered trademark in the United States and other countries licensed exclusively by X/Open Co., Ltd. X/Open is a trademark of X/Open Co. Ltd.

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

ZK5974

This document is available on CD–ROM.

Contents

Preface ix

Part I OpenVMS Computing

1 Introduction to OpenVMS Systems

1.1 What Is the OpenVMS Operating System? 1–1
1.2 OpenVMS Styles of Computing 1–3
1.2.1 Processing Modes and User Interfaces 1–3
1.2.2 Processing Styles 1–4
1.3 Basic OpenVMS Configurations 1–5
1.3.1 Input/Output Connections 1–5
1.3.2 Standalone System Configurations 1–7
1.3.3 VMScluster Configurations 1–7
1.3.4 Networked Systems 1–8
1.4 OpenVMS Systems on Multiple Platforms 1–13
1.4.1 System Compatibility and Program Portability Across Platforms 1–14
1.4.2 Processors on Which OpenVMS Systems Run 1–14
1.5 OpenVMS Growth Potential 1–15

2 OpenVMS Computing Capabilities

2.1 Open System Capabilities 2–1
2.1.1 What Is an Open System? 2–1
2.1.2 Support of Open Standards and Specifications on OpenVMS Systems 2–2
2.1.3 Application Portability 2–3
2.1.3.1 POSIX for OpenVMS Application Portability 2–3
2.1.3.2 Other Application Portability Features 2–4
2.1.3.3 OSF/Motif Applications 2–4
2.1.4 User Portability 2–4
2.1.4.1 OSF/Motif User Interface 2–5
2.1.4.2 POSIX User Interface 2–5
2.1.5 Multivendor Interoperability 2–5
2.1.5.1 Open Networking Capability 2–5
2.1.5.2 Middleware Support for Open Systems 2–6
2.1.5.3 DCE Services Support for Interoperable Applications 2–6
2.2 Distributed Computing Capabilities 2–6
2.2.1 Client/Server Style of Computing 2–6
2.2.2 OpenVMS Client/Server Capabilities 2–7
2.2.2.1 OpenVMS Servers with OpenVMS Clients 2–7
2.2.2.2 OpenVMS Servers with PC Clients 2–7
2.2.3 Middleware in Distributed Environments 2–9
2.2.4 Distributed Software That Is Compliant with OSF Standards 2–9

2.2.5 Distributed Networking Capabilities 2–10
2.2.6 Distributed Features of DECwindows Motif 2–10
2.3 High-Integrity Production System Capabilities 2–11
2.3.1 Dependable OpenVMS Computing Systems 2–11
2.3.2 Availability Tools 2–12
2.3.3 Data Integrity Tools 2–12
2.3.4 Transaction-Processing Capabilities 2–13
2.3.5 Manageability and Security for Data Centers 2–13
2.4 System and Network Management Capabilities 2–14
2.4.1 Managing OpenVMS Systems 2–15
2.4.2 Managing Networks 2–15
2.4.3 Managing Integrated Enterprises 2–16

Part II OpenVMS Systems Software

3 Description of OpenVMS System Software

3.1 OpenVMS Operating System Components 3–1
3.1.1 OpenVMS Kernel 3–1
3.1.1.1 Memory Management Subsystem 3–2
3.1.1.2 Process and Time Management Subsystem 3–3
3.1.1.3 I/O Subsystem 3–3
3.1.1.4 Supporting System Services and Facilities 3–4
3.1.2 OpenVMS Core Services 3–4
3.1.3 OpenVMS Utility Programs 3–5
3.1.4 OpenVMS VAX Vector-Processing Capability 3–6
3.1.5 OpenVMS Symmetric Multiprocessing 3–6
3.1.6 Digital Distributed Transaction Management Services 3–6
3.2 OpenVMS System Management Software 3–7
3.2.1 OpenVMS Installation and Configuration 3–8
3.2.2 Management Software for the General OpenVMS Environment 3–9
3.2.2.1 Controlling System Access 3–10
3.2.2.2 Managing Devices and Storage Media 3–11
3.2.2.3 Backing Up the System 3–11
3.2.2.4 Monitoring, Maintaining, and Tuning the System 3–12
3.2.2.5 Managing Batch and Print Queues 3–13
3.2.3 Management Software for Specific Environments 3–14
3.2.4 OpenVMS Management Station 3–15
3.3 OpenVMS System Security 3–16
3.3.1 OpenVMS Security Environment and Standards 3–17
3.3.2 OpenVMS Security Management Software 3–17
3.4 Optional OpenVMS System Integrated Software 3–20
3.4.1 VMScluster Software 3–20
3.4.2 Volume-Shadowing Software 3–22
3.4.3 RMS Journaling Software 3–23

4 Development on OpenVMS Systems

4.1 Common Programming Environment 4–1
4.1.1 Programming to Standards 4–2
4.1.1.1 Common Environment for Writing Code 4–2
4.1.1.2 Common Language Environment 4–2
4.1.2 Developing Portable Programs 4–3
4.2 OpenVMS Programming Software 4–3

4.2.1 Creating Program Source Files 4–4
4.2.2 Creating Object Files 4–5
4.2.3 Creating Runnable Programs 4–7
4.2.4 Testing and Debugging Programs 4–7
4.2.5 Using Other Program Development Utilities 4–8
4.2.6 Managing Software Development Tasks 4–9
4.3 Using Callable System Routines 4–9
4.3.1 Using DECthreads Run-Time Library Routines 4–9
4.3.2 Using OpenVMS Run-Time Library Routines 4–10
4.3.3 Using OpenVMS System Services 4–11
4.3.4 Using OpenVMS Utility Routines 4–13
4.4 POSIX Programming on an OpenVMS System 4–13
4.4.1 POSIX Applications Using OpenVMS Capabilities 4–14
4.4.2 POSIX for OpenVMS Programming Interface 4–15
4.4.3 POSIX Commands and Utilities 4–15
4.4.4 POSIX Real-Time Functions 4–16
4.4.5 POSIX XPG Support 4–17
4.5 Programming User Interfaces 4–17
4.6 Developing Real-Time Applications 4–18
4.7 Digital Software Development Tools 4–18
4.8 Managing Data 4–19
4.8.1 RMS Files and Records 4–19
4.8.2 RMS Utilities 4–19

5 User Interfaces to the OpenVMS System

5.1 OpenVMS Operating System Access 5–1
5.2 OpenVMS User Environment 5–2
5.2.1 DCL Command Language Functions 5–3
5.2.1.1 DCL Command Procedures 5–4
5.2.1.2 OpenVMS Help System 5–4
5.2.2 OpenVMS DECwindows Motif Interface 5–4
5.3 POSIX Environment on an OpenVMS System 5–7
5.4 Forms-Based User Environments 5–8
5.4.1 DECforms Interface 5–8
5.4.2 ALL–IN–1 Office Systems Environment 5–8
5.5 Information Handling on the OpenVMS System 5–9
5.5.1 OpenVMS Files and Directories 5–9
5.5.2 POSIX Files and Directories 5–10
5.5.3 OpenVMS File Manipulation 5–10
5.5.4 Text File Editing and Processing 5–12
5.5.5 Electronic Mail 5–13
5.5.6 Electronic Conferencing and Text Retrieval Facilities 5–13

Part III Open Distributed Computing Environments

6 OpenVMS Systems in Distributed Environments

6.1 OpenVMS Functions Applicable to Distributed Environments 6–1
6.2 OpenVMS Systems in Distributed Heterogeneous Network Environments 6–2
6.2.1 DECnet/OSI Networking Software 6–3
6.2.1.1 DECnet/OSI Features and Software 6–4
6.2.1.2 DECnet Network Management Tasks 6–5
6.2.2 OpenVMS Connections to TCP/IP Networks 6–7
6.2.3 Network Security 6–8
6.2.4 Other Supported Networking Protocols and Products 6–9
6.3 Multivendor Integration Using Middleware 6–9
6.3.1 Using Middleware with Applications 6–10
6.3.2 Middleware Support for Industry Standards 6–11
6.3.3 Middleware Service Categories 6–11
6.3.4 Distributed Computing Environment Software 6–13
6.3.5 Application Development in Multivendor Environments 6–15
6.3.6 Managing Enterprisewide Multivendor Environments 6–17
6.4 OpenVMS Software in Multivendor Client/Server Environments 6–17
6.4.1 VMScluster Servers and OpenVMS Clients 6–18
6.4.2 OpenVMS Servers in PATHWORKS Environments 6–18
6.4.3 PATHWORKS Server and Client Software 6–19
6.4.4 PATHWORKS Network Connectivity 6–20
6.4.5 OpenVMS Services for PATHWORKS Clients 6–22
6.4.5.1 Mail Services for PC Users 6–23
6.4.5.2 VMScluster Access for PC Clients 6–23
6.4.5.3 OpenVMS Management Services for PC Clients 6–23
6.4.6 Other OpenVMS Services for PC Users 6–24

7 OpenVMS Systems in Commercial Environments

7.1 OpenVMS Production Systems 7–1
7.1.1 OpenVMS Distributed Production Servers 7–2
7.1.2 Providing Dependability in Production System Environments 7–2
7.1.2.1 Maintaining High Availability in Production System Environments 7–2
7.1.2.2 Ensuring Data Integrity in Production Systems Environments . . . 7–4
7.1.3 Managing and Monitoring Production System Environments 7–4
7.1.3.1 Storage Management Products 7–7
7.1.3.2 DECram for OpenVMS Device Driver 7–8
7.1.3.3 Performance Management Tools 7–8
7.1.3.4 Optional VMScluster System Management Software 7–9
7.1.3.5 Business Recovery Server 7–10
7.2 Transaction Processing in Multivendor Environments 7–10
7.2.1 Distributed Transaction-Processing Systems 7–11
7.2.2 Distributed Transaction-Processing Monitors 7–12
7.2.3 Transaction-Processing Support by DECdtm Services on OpenVMS Systems 7–13
7.2.4 Other Distributed Transaction-Processing Products 7–14
7.3 Database Processing in Multivendor Environments 7–15
7.3.1 Database Tools Used in a Distributed Environment 7–15
7.3.2 Database Interoperability Software 7–15

A OpenVMS Support for Standards

Index

Figures

1–1 Software Running on an OpenVMS System 1–2
1–2 Mixed-Interconnect VMScluster Configuration 1–9
1–3 Multivendor Network Topology 1–12
2–1 OpenVMS Services to PC Clients 2–8
3–1 OpenVMS Operating System Components 3–2
3–2 Sample OpenVMS Management Station Screen 3–16
4–1 Developing POSIX Applications 4–14
5–1 DECwindows Motif User Interface 5–6
6–1 Integration of DECnet, OSI, and TCP/IP Network Architectures 6–4
7–1 Two-Phase Commit Protocol for a Distributed Transaction 7–14

Tables

1–1 Examples of Buses, Interconnects, and LAN Adapters Supported by OpenVMS Systems 1–6
1–2 Networking Software Products That Run on OpenVMS Systems 1–10
2–1 POSIX Standards Supported by POSIX for OpenVMS 2–3
3–1 Examples of OpenVMS System Utility Programs 3–5
3–2 Examples of OpenVMS Configuration Utilities and Commands 3–9
3–3 Examples of OpenVMS System Management Utilities 3–10
3–4 Examples of OpenVMS Queue Management Commands 3–13
3–5 OpenVMS Security Features 3–18
3–6 VMScluster Software Components 3–21
4–1 OpenVMS Programming Software 4–4
4–2 Compilers, Interpreters, and Assemblers 4–5
4–3 Other OpenVMS Program Development Utilities 4–8
4–4 Groups of OpenVMS Run-Time Library Routines 4–10
4–5 Groups of OpenVMS System Services 4–11
4–6 OpenVMS Utility Routines 4–13
4–7 Complex Utilities Supported by POSIX for OpenVMS 4–16
5–1 Types of Tasks Performed by Commonly Used DCL Commands 5–3
5–2 Commonly Used DECwindows Motif Applications 5–6
5–3 File Operations Performed Using OpenVMS Utilities and DCL Commands 5–10
6–1 Examples of DECnet/OSI Applications for General-User Operations 6–7
6–2 Components of Digital TCP/IP Services for OpenVMS 6–7
6–3 Middleware Services and Frameworks and Their Digital Implementations 6–12
6–4 Digital DCE Product Family Functionality 6–14
6–5 Network Transports Supported by PATHWORKS Software 6–21

7–1 Software Used to Manage Complex Production Systems 7–5
A–1 OpenVMS Support for Industry and International Standards and Specifications A–2

Preface

The *OpenVMS Software Overview* presents an overall picture of OpenVMS software capabilities and computing environments. It describes how the OpenVMS operating system and related optional software function in diverse environments, including open environments in which OpenVMS operating systems are linked to other systems supplied by multiple vendors. The OpenVMS software features described in this overview indicate the variety of ways in which the software can be used, such as in distributed client/server configurations and dependable production systems.

The descriptions of open system software included in the manual conform to the industry-standard definition of an open system that Digital has adopted: an open system supports standard interfaces, services, and formats that allow application programs and data to be moved easily across systems from different vendors.

This manual describes the multiplatform capabilities of the OpenVMS operating system and related (layered) software. It covers the OpenVMS VAX operating system, which runs on the complete family of VAX processors, and the OpenVMS Alpha system, which runs on Alpha processors ranging from personal computers to very large servers. Discussions of VMScluster environments in this manual generally apply both to VAXcluster systems, which include only VAX nodes, and to VMScluster systems, which include at least one Alpha node.

Intended Audience

This document is intended for anyone who is interested in OpenVMS systems, including managers, analysts, developers, programmers, and general users of OpenVMS systems. Readers of this manual need not have any special knowledge of OpenVMS software products.

Document Structure

The *OpenVMS Software Overview* is divided into three parts, comprising seven chapters and one appendix. Each part of the manual presents one aspect of the overall picture of OpenVMS software capabilities.

Part I presents an introduction to the OpenVMS system, its software, and computing capabilities. It provides a brief summary of system capabilities but does not include details of software design.

- Chapter 1 is a general introduction to OpenVMS systems, software, computing styles, and configurations, including multiplatform use. The chapter also discusses OpenVMS growth potential.
- Chapter 2 describes OpenVMS computing capabilities, including open, distributed, and high-integrity capabilities and manageability.

Part II provides a technical overview of the software available on an OpenVMS system that stands alone or is a member of a VMScluster configuration.

- Chapter 3 describes the components in the base OpenVMS operating system.
- Chapter 4 discusses the OpenVMS software used for developing applications.
- Chapter 5 summarizes the user interfaces to the system.

Part III describes the use of OpenVMS systems in environments in which they are connected to systems supplied by Digital or by other vendors.

- Chapter 6 describes how OpenVMS systems fit in open, distributed computing environments that can include software from multiple vendors.
- Chapter 7 describes how OpenVMS systems support high-integrity, distributed, commercial-strength production system environments.

Appendix A lists the standards supported by OpenVMS software.

Related Documents

This manual is an overview that can be read independently of other OpenVMS documentation. It refers to specific OpenVMS manuals related to certain topics. For a complete listing of OpenVMS manuals, see the *Overview of OpenVMS Documentation*, which lists all documentation for the OpenVMS VAX operating system and the OpenVMS Alpha operating system.

The *OpenVMS Software Overview* also discusses software capabilities of optional (layered) software products that run on the OpenVMS VAX and OpenVMS Alpha operating systems. For documentation that describes POSIX for OpenVMS, refer to the POSIX for OpenVMS documentation set. For information about specific PATHWORKS products, see the appropriate PATHWORKS documentation set. For information about other software that runs on OpenVMS operating systems, refer to the appropriate product documentation.

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

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

Reader's Comments

Digital welcomes your comments on this manual.

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

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

How To Order Additional Documentation

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

Telephone and Direct Mail Orders

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

ZK–7654A–GE

Conventions

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

In this manual, every use of 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.

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:

UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
numbers	All numbers in text are assumed to be decimal unless otherwise noted.

Part I

OpenVMS Computing

This part of the manual presents an overall picture of the OpenVMS system.

Chapter 1 introduces the software that runs on the system and the styles of computing that the system supports. It describes basic OpenVMS configurations and the computing platforms and types of processors on which the OpenVMS system runs. The chapter also discusses the growth potential of OpenVMS systems.

Chapter 2 summarizes major kinds of OpenVMS computing capabilities:

- Open system capabilities
- Distributed system capabilities
- Production system capabilities
- System and network management capabilities

1
Introduction to OpenVMS Systems

The OpenVMS operating system is a highly flexible, general-purpose, multiuser system that supports the full range of computing capabilities, providing the high integrity and dependability of commercial-strength systems along with the benefits of open, distributed client/server systems.

OpenVMS operating systems can be integrated with systems from different vendors in open systems computing environments. Digital has ''opened'' the traditional VMS system to support software that conforms to international standards for an open environment. These industry-accepted, open standards specify interfaces and services that permit applications and users to move between systems and allow applications on different systems to operate together.

The OpenVMS operating system configuration includes OpenVMS integrated software, services and routines, applications, and networks. The system supports all styles of computing, from timesharing to real-time processing to transaction processing. OpenVMS systems configured with optional software support distributed computing capabilities and can function as servers in multivendor client/server configurations.

The OpenVMS operating system is designed to provide software compatibility across all the processors on which it runs:

- OpenVMS VAX software runs on the complete family of VAX processors, which support traditional 32-bit processing.
- OpenVMS Alpha software runs on a full series of very high-speed RISC processors that support 64-bit as well as 32-bit processing.

Both VAX and Alpha processors encompass a wide range of computing power: from desktop and deskside workstations, through distributed office systems and data center servers, to extremely high performance enterprisewide servers.

This chapter introduces the OpenVMS operating system, gives a general overview of the system software, and describes the various styles of computing that OpenVMS software supports. It summarizes the basic ways in which OpenVMS software can be configured and connected to other software, and discusses the hardware platforms and processors on which OpenVMS software runs. The chapter also describes the growth potential of OpenVMS systems.

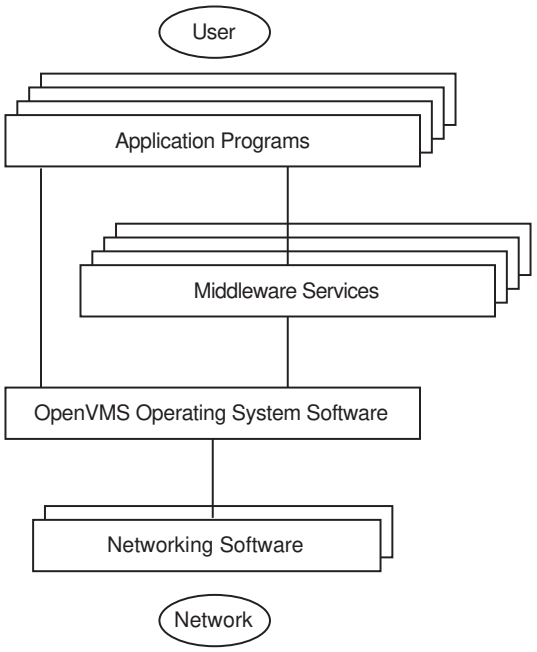
1.1 What Is the OpenVMS Operating System?

The OpenVMS operating system is a group of software programs (or images) that control computing operations. The base operating system is made up of core components and an array of services, routines, utilities, and related software. The OpenVMS operating system serves as the foundation from which all optional software products and applications operate. The services and utilities in the base OpenVMS operating system support functions such as system management, data management, and program development. Other integrated software that adds

value to the system provides functions such as clustering and volume shadowing. (For a description of OpenVMS operating system software, see Chapter 3.)

Optional software products, including application programs developed by OpenVMS programmers and other programmers, run on the core operating system (as shown in Figure 1–1). The OpenVMS system supports a powerful, integrated development environment with a wide selection of software development tools supplied by Digital and other vendors. Application programs written in multiple languages provide computational, data-processing, and transaction-processing capabilities. Thousands of applications have been developed for OpenVMS systems by Digital and independent software vendors.

Figure 1–1 Software Running on an OpenVMS System



ZK–5460A–GE

Networking software permits OpenVMS systems to communicate with other Digital systems and with systems supplied by other vendors in worldwide networks, including the global Internet. Special-purpose networking software enables OpenVMS systems to function as powerful servers to personal computer clients in distributed client/server environments.

Optional software that supports multivendor integration can be layered between the operating system and applications. This optional software is called middleware (see Figure 1–1). Middleware run-time services provide standard application programming interfaces (APIs) to accomplish the interaction between applications and their users, data, systems, and other applications. Digital middleware products, based on international standards, can also run on other standards-compliant systems. These products provide support for applications in distributed, multivendor environments. (For a discussion of how middleware provides open system capability, see Section 2.1.5.2.)

OpenVMS VAX and OpenVMS Alpha software exhibits compatibility from version to version:

- User-mode programs and applications created under earlier versions of OpenVMS VAX or OpenVMS Alpha run under subsequent versions with no change.
- Command procedures written under one version of OpenVMS continue to run under newer versions of the software.

OpenVMS software developed on VAX platforms can migrate easily to Alpha platforms (see Section 1.4.1):

- Most OpenVMS VAX images run under translation on OpenVMS Alpha systems.
- Most user-mode OpenVMS VAX sources can be recompiled, relinked, and run on an OpenVMS Alpha system without modification. Code that explicitly relies on the VAX architecture requires modification.

1.2 OpenVMS Styles of Computing

The OpenVMS operating system offers flexible, reliable support for all styles of computing. OpenVMS provides concurrent support for traditional modes of processing (batch, interactive, real-time) and multiple user interfaces (keyboard, windowing, and forms-based). In addition, OpenVMS can be easily tuned to support applications involving different computing styles: for example, timesharing, multiprocessing, and distributed client/server computing.

This section describes the varied styles of computing available to OpenVMS users. Chapter 2 summarizes computing capabilities supported by OpenVMS, including open, distributed, and commercially oriented capabilities. Chapter 6 and Chapter 7 describe the software that runs on OpenVMS systems in distributed multivendor environments.

1.2.1 Processing Modes and User Interfaces

In the traditional interactive mode of communication with the OpenVMS operating system, the user enters a command, and the system executes it and responds. In the batch mode, commands to be executed by the operating system are placed in a file and submitted to the operating system for execution.

Real-time processes do not have to compete with interactive or batch jobs for scheduling priority within the OpenVMS operating system. A real-time process responds to events in related processes as they occur, rather than when the computer is ready to respond to them. Scheduling services allow system managers to provide their own scheduling algorithms.

Standard user interfaces to the OpenVMS operating system include the keyboard interface and the windowing interface. Users with traditional character-cell terminals can interact with the keyboard interface to the system. Users with workstation terminals can access an optional graphical windowing interface, the DECwindows Motif interface. The user can employ point-and-click techniques to interact with menus and dialog boxes on multiple windows on the screen. DECwindows users can also invoke a window that emulates an OpenVMS terminal and can interact with the keyboard interface. Another possible user interface is a transaction interface, in which the user responds to a menu-driven forms interface customized for a particular application, such as an office application. Chapter 5 describes the various OpenVMS user interfaces.

Additionally, end users on personal computers can access shareable information and resources on OpenVMS systems through special PATHWORKS software (as described in Section 2.2.2.2).

1.2.2 Processing Styles

An OpenVMS system can be a single-user system or can provide timesharing functions for multiple users. In a timesharing configuration, many people can use the same computer at once. The computer interleaves time intervals for different user programs, performing a small portion of one user's program and then shifting to the next user. OpenVMS timesharing software is described in Section 3.1.1.

An OpenVMS system can be a uniprocessing or multiprocessing system. A uniprocessing system comprises a single central processing unit (CPU) executing one copy of the OpenVMS operating system code and running one file system independent of other CPUs. In OpenVMS multiprocessing implementations, two or more processors are coupled together. Multiprocessing configurations include the following:

- The OpenVMS symmetric multiprocessing system (SMP) is a tightly coupled system in which all processors run the same copy of the operating system in memory. All processors can execute code from a single shared memory address space, dividing and sharing the workload. The processors are adjacent to each other, boot and shut down together, run a single file system, and appear as a single system from the system management and user points of view. OpenVMS SMP is described in Section 3.1.5.
- A VMScluster system consists of a group of connected CPUs,[1] each containing one or more processors. CPUs that are members of a VMScluster system can share processing, mass storage, and other resources under a single management and security domain. Within this highly integrated environment, members retain their independence because they use local, memory-resident copies of the OpenVMS operating system. VMScluster members can boot and shut down independently while benefiting from common resources. The VMScluster is perceived by outside systems in the network as being a single system entity. VMScluster configurations are described in Section 1.3.3 and VMScluster software components in Section 3.4.1.
- A network is an example of a loosely coupled configuration. In a network, OpenVMS systems, VMScluster systems, and other systems can be linked together, but the systems (nodes) can be separate, can be managed independently, and can run separate file systems. Networked systems are described in Section 1.3.4.

OpenVMS supports centralized and distributed processing capabilities across the full range of processors. Centralized processing involves having a single system control the use of computing resources and the execution of applications. OpenVMS provides the capability of controlling processors and their associated resources in a distributed manner as well as a centralized manner.

[1] VMSclusters include Alpha CPUs and, optionally, VAX CPUs. Descriptions of VMSclusters also apply to VAXclusters composed of VAX CPUs.

Distributed processing is the style of computing that permits portions of an application to be run on different systems, yet appear to the user as one integrated environment. The distributed application program can be divided into independent subsystems or modules that execute on different systems. The data and code for each module are separate, but the modules communicate with each other to share data or access to files or databases.

An example of one style of distributed computing is client/server computing: a client portion of an application on one system requests services from the server, which is usually on a different system.

For a description of OpenVMS distributed processing capabilities, see Section 2.2. Distributed computing environments are covered in Chapter 6.

1.3 Basic OpenVMS Configurations

OpenVMS systems are highly flexible and support a great variety of interconnections and configurations, including the standalone OpenVMS configuration, VMScluster configurations, and network configurations involving OpenVMS systems. OpenVMS interconnections and configurations are described in the following sections.

1.3.1 Input/Output Connections

OpenVMS input/output (I/O) software supports connections to peripherals, communications devices, and networks. The I/O subsystem, which is part of the OpenVMS operating system kernel, processes I/O requests to external devices (as described in Section 3.1.1).

OpenVMS I/O provides access to storage buses and interconnects and system-dependent open buses. (Open buses are those for which any vendor can obtain specifications and build adapters.) Examples of equipment that can be attached to buses and interconnects supported by OpenVMS systems include conventional disk and tape devices, compact disc devices, other terminal and workstation connectors, and communications adapters. Other devices supplied by users can also be attached to OpenVMS buses and interconnects.

OpenVMS I/O provides for connection to other systems, devices, and local area networks (LANs) by means of LAN adapters. The I/O subsystem also supports connection to protocols that permit communication with other systems over DECnet and TCP/IP networks (as described in Section 1.3.4).

Table 1–1 lists examples of some of the buses, interconnect devices, and LAN adapters supported by OpenVMS I/O subsystems.

Table 1–1 Examples of Buses, Interconnects, and LAN Adapters Supported by OpenVMS Systems

Device	Description
Storage Buses and Interconnects	
CI computer interconnect bus	Very high-speed, dual-path bus that joins computers and intelligent I/O subsystems.
Digital Storage System Interconnect (DSSI) bus	High-speed storage bus that permits multiple computers to communicate directly with storage devices.
Small Computer System Interface (SCSI)	Interface that permits devices complying with the ANSI SCSI standard to be connected to a host processor or computer. The SCSI interconnect can be used as a storage interconnect for multiple Alpha systems in a VMScluster system.
Open Buses and Interconnects	
OpenVMS SCSI implementation	Open interconnect that supports SCSI-compliant devices supplied by other vendors and user-written device drivers.
PCI	Personal computer interface, an open industry-standard interface used by the PC industry and by 64-bit Alpha processors.
TURBOchannel I/O interconnect	Open bus for desktop applications, used on desktop workstation systems; provides the high performance required to support advanced computing systems involving complicated I/O options (for graphics, imaging, high-speed networking and data collection, and large database storage). TURBOchannel is a Digital open standard; the specifications are available to other vendor designers.
VME bus	High-performance open I/O bus that supports device drivers written by other vendors and can act as a system interconnect from multiprocessing systems.
LAN Adapters	
Ethernet (IEEE 802.3) adapter	Communications device that provides a reliable, low-cost, coaxial cable connection.
Fiber Distributed Data Interface (FDDI) adapter	Communications device that provides a connection to a high-performance fiber-optic interconnect.
Token ring (IEEE 802.5) LAN adapter	Communications device that provides a connection to a baseband token ring LAN.

For communication with personal computers and Macintosh systems in a PATHWORKS environment (described in Section 2.2.2.2), OpenVMS servers can connect to Ethernet or token ring LANs. For communication with devices such as terminals, modems, and printers offered by terminal servers on a LAN, OpenVMS systems support connection to the LAT local area transport software. OpenVMS also supports connection to the local area disk (LAD) protocol for access to compact disc media (CD–ROMs) that reside on Digital InfoServer systems. In addition, the I/O software permits direct and remote connection of monitors, terminals, and windowing workstations.

1.3.2 Standalone System Configurations

A standalone OpenVMS system operates independently, not linked to other systems. The basic hardware configuration comprises the following:

- The CPU: The hardware that handles all computing operations, calculations, routing of input and output, and execution of programs. A standalone system can be a uniprocessor system supporting one processor or a multiprocessor system supporting two or more processors.
- The memory modules: A series of physical locations that store data and instructions in binary form.
- I/O: Device drivers (a combination of hardware and software), peripheral devices, and buses and interconnects.

The OpenVMS software running on the standalone hardware platform can handle timesharing activities for multiple users as well as execute multiple individual programs for single users. OpenVMS standalone capabilities are described in Part II of this manual. Section 6.1 compares capabilities supported by OpenVMS in standalone and distributed environments.

1.3.3 VMScluster Configurations

The ability to survive failure of any single component makes the VMScluster system suitable for developing high-availability applications such as transaction-processing applications. VMSclusters also make effective servers for clients on personal computers and other desktop systems (see Section 2.2.2.2).

A VMScluster system is a highly integrated OpenVMS computing environment. A VMScluster system includes a connected group of Alpha computers or, optionally, a combination of Alpha and VAX computers that share resources under a single OpenVMS management domain. A VMScluster system that includes at least one VAX system is called a dual-architecture VMScluster configuration.[2] The traditional VAXcluster system includes only VAX systems. Descriptions of VMScluster environments also apply to VAXcluster systems, unless indicated otherwise.

In VMScluster environments, shared resources include image and data files, batch queues, print queues, disk and tape storage, and system management. To users and applications, the VMScluster system functions as a single system, except that applications and data continue to be available to users on remaining members (nodes) of the VMScluster system even if a member shuts down. Within the VMScluster configuration, each member can boot or shut down independently. If one member shuts down, the user can log in to another member of the cluster and continue working. Because disk and tape storage is shared across the VMScluster, the user can continue to access the original data and applications. Individual members of the VMScluster system can be serviced without interruption to applications running on the VMScluster.

VMScluster systems can make disk and tape resources available to cluster nodes. Through disk and tape server software on the VMScluster, cluster-accessible disks and tapes can be accessed simultaneously by multiple active nodes in the VMScluster. Cluster-accessible disks permit high-performance, clusterwide read/write file sharing. Because computers can share a single version of a file, updates to a file need be made only once. Clusterwide queues permit sharing of

[2] Refer to the appropriate VMScluster Software Product Description for the version numbers of OpenVMS VAX and OpenVMS Alpha systems supported.

batch processing and printer resources. Batch and print queues can be processed on any node that has access to the disk.

The VMScluster system is a combination of hardware and software. VMScluster hardware includes a variety of buses and other connection devices as well as controllers for resource sharing. Hardware resources can be expanded incrementally or gradually, as necessary. VMScluster software includes drivers, controllers, and traffic managers (as described in Section 3.4.1). VMScluster processors, controllers, disks, and tapes can be added or removed without requiring that the VMScluster be shut down.

VMScluster configurations are highly flexible and support the full range of Alpha and VAX processors with the OpenVMS operating system. In a VMScluster system, two or more CPUs are connected by means of communications media, called interconnects, which the VMScluster software uses for System Communications Architecture (SCA) I/O traffic. The VMScluster members use interconnects (as described in Table 1–1) to communicate with each other. In a VMScluster, Alpha and VAX nodes can use any combination of the CI, DSSI, Ethernet, and FDDI interconnects; Alpha nodes can also use the SCSI interconnect.

A VMScluster system that uses more than one type of interconnect for VMScluster communication is referred to as a mixed-interconnect VMScluster system. An example of a mixed-interconnect VMScluster configuration is given in Figure 1–2.

A VMScluster configuration that includes a CI interconnect can optionally be configured with HSC hierarchical storage controller units that let nodes connected to the CI share disks. Each member of a multihost DSSI VMScluster can share all disks attached to the DSSI bus.

Ethernet and FDDI are industry-standard general-purpose communication interconnects that may be used to implement a LAN. A VMScluster system that uses a LAN interconnect is called a local area VMScluster.

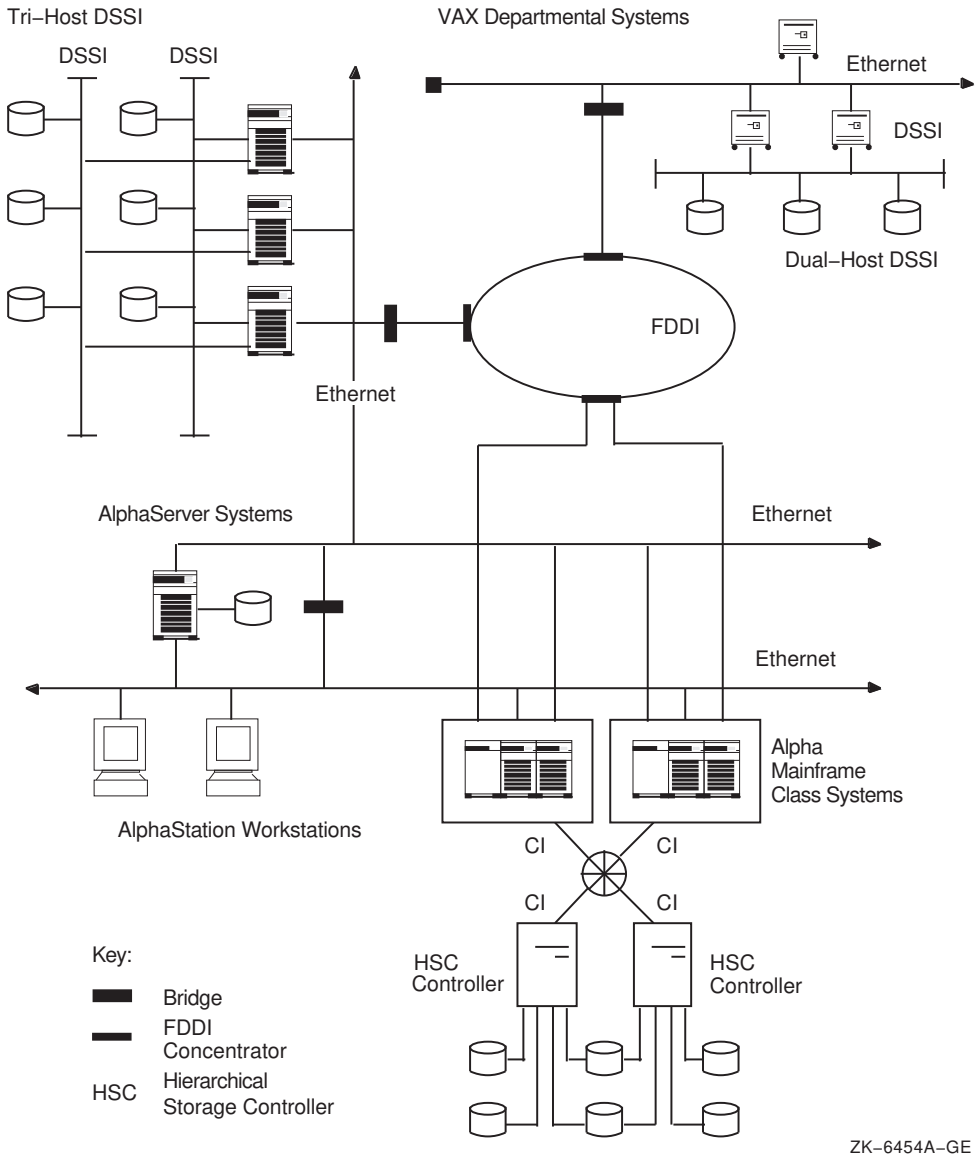
In a multisite VMScluster system, member nodes are located at geographically separate sites. Multisite VMSclusters are configured using wide-area ATM (asynchronous transfer mode) technology or DS3 technology (also called T3), in conjunction with FDDI LAN technology and appropriate bridging. The ATM and DS3 communications services, available from common telephone service carriers and other sources, provide long-distance, point-to-point communications.

The VMScluster software for OpenVMS Alpha provides for configurations of single- or multiple-host SCSI buses that support SCSI storage devices, controllers, and support components.

1.3.4 Networked Systems

Networking software and hardware products permit OpenVMS operating systems to communicate with other systems. The network consists of computer systems connected by physical means, such as cables or microwave links. Optional networking software running on the OpenVMS system supports networking protocols (sets of rules and operating procedures) that permit communication between Digital and other vendor systems in worldwide networks.

Figure 1–2 Mixed-Interconnect VMScluster Configuration



DECnet and TCP/IP networking products that run on OpenVMS VAX and Alpha systems are listed in Table 1–2. In addition, DECnet, TCP/IP, and AppleTalk network transports permit OpenVMS servers to communicate with personal computer and Macintosh clients in PATHWORKS environments. Use of networking products in multivendor distributed computing environments is summarized in Chapter 6.

Table 1–2 Networking Software Products That Run on OpenVMS Systems

Digital Networking Product	Functions
DECnet Software	
DECnet/OSI for OpenVMS	Supports OSI protocols, permitting communication with any vendor's system that supports standards for Open Systems Interconnection (OSI)
	Supports DNA protocols, permitting communication with any system running DECnet/OSI or DECnet Phase IV software on the DECnet network
	Supports running DECnet and OSI applications over TCP/IP transports† to communicate with systems based on UNIX
DECnet for OpenVMS	Supports DECnet Phase IV DNA protocols, permitting communication with any system running DECnet Phase IV software on the DECnet network
TCP/IP Software	
Digital TCP/IP Services for OpenVMS	Supports the TCP/IP protocol, permitting communication with systems based on UNIX and other systems on the Internet
NFS software‡	Supports the NFS network file system protocol: the NFS server software permits UNIX clients to access OpenVMS files and files compatible with UNIX on OpenVMS systems, and the NFS client software permits OpenVMS users to access remote NFS files

†Requires a separate TCP/IP product on the same system
‡Components of Digital TCP/IP Services for OpenVMS

DECnet is the traditional Digital networking product, supported on most Digital systems, including the OpenVMS operating system and operating systems based on UNIX (such as Digital UNIX). DECnet software provides peer-to-peer networking. DECnet software on the OpenVMS system has proven its reliability over the years in extensive networking configurations, including the very large Digital network.

DECnet for OpenVMS VAX (previously known as DECnet–VAX Phase IV), uses DNA protocols (that conform to the Digital Network Architecture) to communicate with other DECnet systems. The Phase IV product that runs on OpenVMS Alpha is called DECnet for OpenVMS Alpha.

DECnet/OSI (the latest phase of DECnet) conforms to international standards for the Open Systems Interconnection (OSI) model (as described in Section 2.1.5.1) and supplies open, multiprotocol, multivendor networking capabilities. DECnet/OSI for OpenVMS software, which can run on VAX and Alpha platforms, integrates OSI and DECnet Phase IV capabilities, includes the ability to use TCP/IP transports, and supports additional networking capabilities. Routing is provided by multiprotocol routers.

DECnet/OSI is compatible with Phase IV, the previous phase of DECnet. This compatibility between phases preserves the investment in networking capabilities made by DECnet users on OpenVMS systems.

The Digital TCP/IP Services for OpenVMS product (which is the latest version of the VMS/ULTRIX Connection product) supports resource sharing between OpenVMS systems, UNIX systems, and other systems that support the TCP/IP protocol suite and NFS (see Section 6.2.2). TCP/IP, the de facto standard for networking UNIX systems, is used by the Internet, an openly accessible, worldwide research and commercial network. OpenVMS systems running TCP/IP software supplied by Digital or other vendors can access the Internet.

DECnet/OSI can support a multivendor, multiprotocol network environment when configured with the Digital TCP/IP Services for OpenVMS product or another TCP/IP product running on the same OpenVMS system. DECnet applications can be run over DNA, OSI, or TCP/IP transports. OSI applications can be run over OSI or TCP/IP transports.

Digital networking systems can be connected to local area networks (LANs) or wide area networks (WANs). A LAN is a network of systems in a specific geographical area, while a WAN permits long-distance communications. The two kinds of networks can be integrated into a single network.

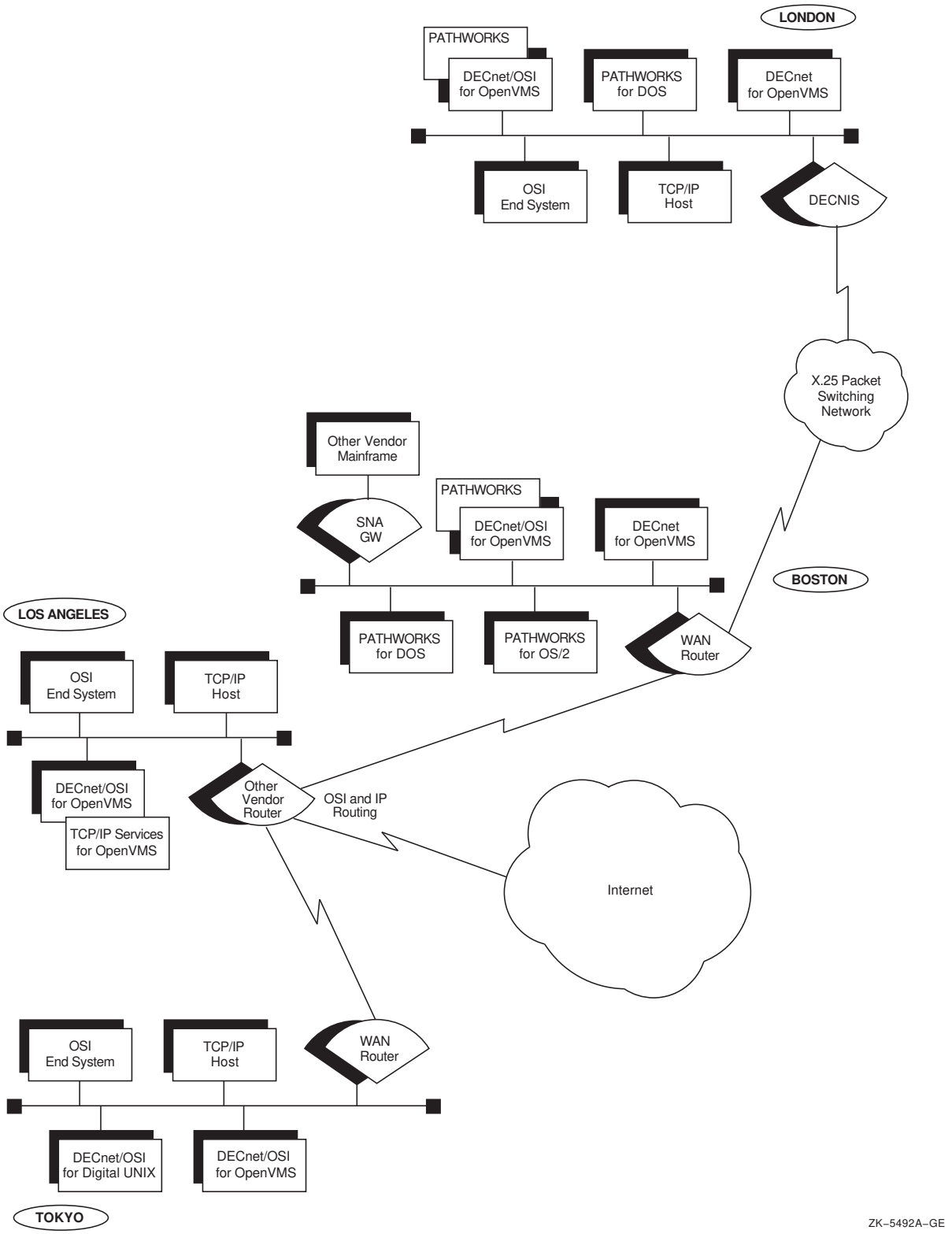
High-speed LAN communications occur over Ethernet/IEEE 802.3 or FDDI interconnections. Ethernet networks are flexible, reliable, and inexpensive. The FDDI fiber optic interconnect provides 10 times the transmission rate of the Ethernet and can seamlessly interoperate with an Ethernet on a LAN extended by means of a LAN bridge. A third LAN technology supported by OpenVMS is the IEEE 802.5 standard token-passing ring. The token ring LAN is a baseband ring network compatible with IBM 802.5/token ring networks. Ethernet and token ring LAN technologies are supported in PATHWORKS configurations.

DECnet/OSI supports connections to X.25 packet-switching data networks (for example, TYMNET) that conform to international recommendations. DECnet also supports gateway software for connection to other networking products (for example, IBM SNA interconnect products).

OpenVMS systems can also act as servers to personal computer clients (such as DOS, Windows, OS/2, and Macintosh clients) over network connections, using appropriate PATHWORKS software on each client and server. OpenVMS servers support Ethernet or token ring LAN connections, TCP/IP connections, and AppleTalk connections. The PATHWORKS environment is introduced in Section 2.2.2.2 and described in detail in Section 6.4.2.

Figure 1–3 shows a multivendor open network that includes DECnet/OSI, DECnet Phase IV, and TCP/IP systems as well as systems provided by other vendors. Routing is provided by multiprotocol wide area networking routers, including the DEC WANrouter, the DECNIS network integration service router, and other vendor multiprotocol routers.

Figure 1–3 Multivendor Network Topology



1.4 OpenVMS Systems on Multiple Platforms

The OpenVMS operating system is available on two hardware platforms:

- A complex instruction set computer (CISC) architecture based on the VAX architecture
- A reduced instruction set computer (RISC) architecture based on the Alpha architecture

The OpenVMS VAX operating system runs on the CISC platform, and the OpenVMS Alpha operating system runs on the RISC platform. The OpenVMS Alpha operating system is a port of the OpenVMS VAX operating system, not a redesign of the operating system.

A computing architecture provides a set of rules that determine how the computer actually works; for example, the architecture defines how the data is addressed and the instructions are handled.

The VAX architecture is a flexible CISC architecture that provides for 32-bit processing. The instruction set consists of a large number of instructions of variable length, some of which are complex. The 32-bit VAX architecture provides sufficient address space for current and future applications. The capabilities of OpenVMS on VAX processors will continue to be expanded.

The VAX architecture is used across the entire VAX family of systems. The use of a single, unifying, integrated VAX architecture with the OpenVMS software permits an application to be developed on a VAX workstation, prototyped on a small VAX system, and put into production on a large VAX processor. The advantage of the VAX system approach is that it enables individual solutions to be tailored and fitted easily into a larger, enterprisewide solution.

The Alpha architecture is a high-performance RISC architecture that can provide 64-bit processing on a single chip. The use of the RISC architecture in Alpha processors offers increased CPU performance. The instruction set consists of uniformly sized simple instructions that can be executed rapidly. The 64-bit capability is especially useful for applications that require high performance and very large addressing capacity.

The hardware designs of VAX and Alpha processors are inherently reliable and are particularly suitable for high-availability applications (such as dependable applications for mission-critical business operations and server applications for a wide variety of distributed client/server environments).

A move to the Alpha design from a VAX processor can take place whenever applications require the performance or capacity. Alpha processor support for the 64-bit addressing capability provides sufficient primary data storage for the most data-intensive applications and for applications that perform very large numbers of high-speed I/O operations. For example, Alpha processors are appropriate for graphics- or numeric-intensive software applications that involve imaging, multimedia, visualization, simulation, and modeling.

1.4.1 System Compatibility and Program Portability Across Platforms

The OpenVMS Alpha operating system is compatible with OpenVMS VAX systems in terms of user, system manager, and programmer environments. For general users and system managers, OpenVMS Alpha has the same interfaces as OpenVMS VAX. Virtually all OpenVMS VAX system management utilities, command formats, and tasks are identical in the OpenVMS Alpha environment. Additionally, OpenVMS VAX and OpenVMS Alpha systems can coexist in the same networking environment and in the same VMScluster system.

For programmers, the OpenVMS Alpha software development environment involves the same programming interfaces and development tools as with OpenVMS VAX. Most OpenVMS VAX application programs, especially those written in higher level languages, can be recompiled and relinked and then run on OpenVMS Alpha. Alternatively, if recompilation is not practical or possible, the DECmigrate for OpenVMS AXP layered product can be used to translate user-mode OpenVMS VAX images to run on OpenVMS Alpha systems (see Section 4.2.2).

Software compatibility on the two platforms and considerations involved in developing portable programs that can run on both OpenVMS VAX and OpenVMS Alpha systems are discussed in *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*.

For current information about available OpenVMS VAX and OpenVMS Alpha capabilities, refer to the applicable Release Notes and New Features manuals for your system.

1.4.2 Processors on Which OpenVMS Systems Run

OpenVMS runs on the full range of VAX and Alpha processors:[3]

- VAX processor-based systems supported include desktop and deskside workstations and workgroup, departmental, and data center servers.
- Alpha processor-based systems supported include desktop and deskside workstations, personal computing systems, and workgroup, departmental, and data center servers.

OpenVMS can run on VAX processors that range in scale from VAXstations and MicroVAX systems, step by step, through the largest of the VAX based systems, such as the VAX 7000 models. At any time a user may migrate from the VAX based systems to the Alpha family, which offers higher performance systems in configurations similar to the VAX family.

Alpha processors range from low-end client workstations through extremely fast servers to powerful, high-capacity enterprise servers. The AlphaStation systems and AlphaServer systems provide 64-bit processing capability on high-performance PCI buses (which are standard in the PC industry). Multiple operating systems, including Digital UNIX and Windows NT systems as well as OpenVMS systems, can run on Alpha workstations and servers.

OpenVMS also promotes growth by supporting both symmetric multiprocessing (SMP) and VMScluster systems. SMP provides for an easy way to increase system performance without disrupting business operations. SMP support is available when it is included in the hardware design. VMSclusters ensure

[3] For a current list of processors supported by an OpenVMS system, refer to the applicable OpenVMS Software Product Description.

increased system availability for mission-critical applications. Clustering support is provided across the full range of both VAX and Alpha families.

OpenVMS, combined with the industry-benchmark VAX processor and high-performance Alpha processor, provides continuity of systems from the desktop to the data center. These systems provide environments for industry-standard personal software, engineering-design work, and business solutions ranging from supporting the one-person office to the very largest corporations.

1.5 OpenVMS Growth Potential

OpenVMS managers and users can expand and alter their computing capabilities without risk of obsolescence or system-related constraints. In addition to being scalable for the full range of processors from desktops to mainframes, the OpenVMS operating system offers the flexibility to move the software from VAX platforms to Alpha platforms as processing requirements grow or change. Moreover, OpenVMS distributed multivendor system capabilities give managers the opportunity to integrate systems and devices from multiple vendors into the OpenVMS computing environment and to integrate OpenVMS software into other vendors' environments.

OpenVMS protects computing resources against obsolescence and incompatibility and ensures investments in applications and data across systems. The system also supports new technologies and standards as they are developed without sacrificing any existing capabilities. Over the years, new capabilities such as VAXcluster systems, multiprocessing, and disaster tolerance have been integrated into the OpenVMS environment. All the while, programs compiled and linked on the earliest VAX processors run, without change, on the newest VAX processor, using the latest version of the OpenVMS operating system; these same programs can also migrate easily to run on Alpha processors.

OpenVMS VAX and OpenVMS Alpha systems are functionally equivalent. The OpenVMS Alpha environment supports the same full software capability that is available with OpenVMS VAX systems. The compatibility of the OpenVMS operating system on all processors protects the investment in application development and training. It is possible to add capacity without having to revise applications or retrain people.

OpenVMS offers multiple growth options. OpenVMS customers can upgrade or replace existing software, group systems into VMSclusters or add to existing VMScluster configurations, or link systems through networking connections.

Upgrades of processor hardware do not involve a long, complicated system generation process. The OpenVMS operating system boots on the processor or workstation, automatically configuring itself to the CPU, memory, and devices present. Installation of the OpenVMS Alpha software and layered products is simplified through the use of the POLYCENTER Software Installation utility. (The traditional OpenVMS installation procedure is used to install OpenVMS VAX systems.) The user can upgrade to new software releases or add layered software with minimal disruption.

OpenVMS is highly flexible and can accommodate to change readily. As the configuration of the OpenVMS system is modified through the addition or subtraction of systems or changes in vendors' devices, the OpenVMS system will continue to work in a reliable, predictable manner. VMScluster design permits the addition of multiple processors, storage controllers, disks, and tapes without requiring that the VMScluster system be shut down. OpenVMS distributed computing capabilities permit systems and resources to be added

without disruption of the network. In addition, with OpenVMS support for open interfaces, managers can determine what open systems and other capabilities are required to meet their needs. OpenVMS standards-based open networking capabilities permit free interconnection of systems from different vendors. Using Digital middleware, applications can move easily across different systems in a network and have the systems all work together as if they came from one vendor.

2
OpenVMS Computing Capabilities

The OpenVMS operating system provides an array of capabilities that support the full range of computing environments. A computing environment is made up of resources that are compatible with each other and all work together toward a common goal. In general, OpenVMS environments can supply the following kinds of capabilities (which can exist in any combination):

- Open system capabilities
- Distributed processing capabilities
- Production system capabilities
- System and network management capabilities

OpenVMS software capabilities include both the standardized features of open systems computing and the commercial-strength functionality of traditional OpenVMS systems. System and network management software provides for control of heterogeneous, integrated environments.

This chapter describes the capabilities supported in OpenVMS computing environments and summarizes the software resources available in each kind of environment.

2.1 Open System Capabilities

OpenVMS offers the benefits of an open system environment, which permits both applications and users to move between systems. In addition, applications on different open systems can operate together.

The OpenVMS operating system makes available a set of services in an open domain, while still offering its traditional high-integrity computing services. Incorporation of open computing capabilities enhances the traditional feature-rich OpenVMS environment.

2.1.1 What Is an Open System?

The Institute of Electrical and Electronics Engineers (IEEE) Computer Society has developed a definition of an open system that Digital has adopted. This definition is based on standard interfaces rather than on standardization of products. The IEEE definition of an open system is as follows:

> A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software:
>
> - To be *ported* across a wide range of systems with minimal changes
> - To *interoperate* with other applications on local and remote systems, and
> - To *interact* with users in a style which facilitates user portability.

The IEEE definition uses the term ‘‘open specification’’: ‘‘A public specification that is maintained by an open, public consensus process to accommodate new technology over time and that is consistent with standards.’’

Open specifications do not rely on any particular technology or product; they allow users to determine what open systems and other capabilities are required to meet their needs.

Software in the OpenVMS open systems environment enables the development and use of portable applications and consistent user interfaces and also permits systems to operate together. The keys to openness of OpenVMS systems are standard programming interfaces, standardized user interfaces, and standard protocols.

2.1.2 Support of Open Standards and Specifications on OpenVMS Systems

Digital is active in standards bodies that specify or adopt critical standards for an open environment. As part of this effort, Digital has extended VMS to conform to the majority of open, international standards for portability, interoperability, and a consistent user interface. Some of the open standards and specifications supported by OpenVMS are:

- IEEE standards for POSIX, which is the portable operating system interface
- X/Open standards according to the X/Open Portability Guide Issue 3 (XPG3) BASE specification and Issue 4 (XPG4) BASE profile specifications
- Standards developed by the Open Software Foundation (OSF) for:
  - Application Environment Specification (AES)
  - Distributed Computing Environment (DCE)
- The International Organization for Standardization (ISO) standards for the Open Systems Interconnection (OSI) model
- American National Standards Institute (ANSI) accredited standards for languages [1]
- Internet Engineering Task Force (IETF) Request for Comments (RFCs)

Additionally, Digital participates in the workshops of the National Institute of Standards and Technology (NIST, formerly NBS) of the U.S. Government and supports NIST’s work in defining an environment, based on formal standards, in which to develop and support portable applications. Digital is also certified by NIST as an accredited U.S. Government OSI Profile (GOSIP) Test Facility.

POSIX for OpenVMS has passed the NIST-developed POSIX Conformance Test Suite (PCTS) and has been granted a certificate of validation for conformance to Federal Information Processing Standards Publication (FIPS) 151-1 and 151-2 by NIST.

The OpenVMS operating system has received X/Open BASE-level branding for the X/Open Portability Guide Issue 3 (XPG3) environment. OpenVMS Alpha has also received XPG4 BASE profile branding for the X/Open Portability Guide Issue 4 (XPG4), which extends the internationalization features of the guide. The X/Open Portability Guide is specified by the X/Open consortium, a consortium of major vendors that is establishing a Common Applications Environment based on formal standards and other widely accepted specifications.

[1] Note that some ANSI standards are technically identical to ISO standards, such as the ISO/ANSI C standard.

For a summary of standards and specifications that OpenVMS supports, refer to Appendix A.

2.1.3 Application Portability

Application portability is the capability to easily move an application from one system to another. Standard programming interfaces permit application and data portability. Portable applications written strictly to a suite of open specifications provide the following benefits:

- Applications can be written once and run on other open platforms that support the standards used in the applications.
- Users can access the wide range of applications available on open platforms.
- Applications can be supplied by different vendors.

Applications that are developed on the OpenVMS VAX or OpenVMS Alpha system and conform to open standards can be easily ported to other systems that conform to the same standard interfaces. Applications written in ISO and ANSI languages are portable to other systems. In addition, the OSF/Motif graphical user interface supports application portability.

2.1.3.1 POSIX for OpenVMS Application Portability

POSIX standards enable users and conforming applications to move between systems supporting the POSIX standards. OpenVMS VAX and Alpha systems support POSIX and X/Open standards and draft standards, thereby allowing portable applications to benefit from the proven features of OpenVMS.

IEEE POSIX standards are a set of implementation-independent system interface standards. IEEE POSIX standards have the ''look and feel'' of UNIX interfaces. Table 2–1 lists the standards supported by POSIX for OpenVMS.

Table 2–1 POSIX Standards Supported by POSIX for OpenVMS

POSIX Standard or Draft Standard	Description
1003.1 (standard)	Specifies standard operating system interface and environment, in terms of callable functions, data structures, and header files, using the C language
1003.1a (draft standard)	Specifies the system interface extension
1003.1b† (standard)	Specifies extensions for developing real-time applications and extends the services provided in 1003.1
1003.2 (standard)	Specifies utilities and commands built into the shell; also specifies a number of callable functions (relating to regular expression handling); includes user portability extensions (previously specified in POSIX 1003.2a)

†Renumbered from 1003.4

The availability of POSIX 1003.1b (previously 1003.4) and 1003.2 distinguishes POSIX for OpenVMS from many other proprietary operating systems, which often limit their standards offering to 1003.1.

POSIX for OpenVMS supports X/Open standards specified in the XPG3 BASE specification and XPG4 BASE profile specifications.

OpenVMS has received the XPG3 BASE brand, which confirms that the OpenVMS system provides the open systems functionality specified by XPG3. XPG3 BASE-level branding applies to the operating system and covers:

- System interfaces (includes POSIX 1003.1 plus extensions)
- Commands and utilities: A shell environment (includes POSIX 1003.2 with extensions)
- Internationalization features

OpenVMS has also received the XPG4 BASE profile brand, which covers internationalized system calls and libraries, plus wide character functions, commands, utilities, and the C language as defined by XPG4.

An application that strictly conforms to POSIX and X/Open standards can be developed on an OpenVMS system with POSIX for OpenVMS and then ported without modification to any other platform that supports the same POSIX and X/Open standards or draft standards. Similarly, an application that strictly conforms to POSIX standards that is developed on a platform other than OpenVMS can be ported to and run on an OpenVMS system on which POSIX for OpenVMS is installed. (POSIX for OpenVMS programming is described in Section 4.4 and POSIX for OpenVMS user interfaces in Section 5.3.)

POSIX applications and users on OpenVMS can access and benefit transparently from the feature-rich OpenVMS environment in addition to the standard interfaces. Some of the key benefits directly available for POSIX users on OpenVMS are OpenVMS security features, volume shadowing, clustering, and many features of the software development environment. The availability and use of POSIX for OpenVMS does not affect other regular OpenVMS users.

2.1.3.2 Other Application Portability Features

Applications written in ISO/ANSI languages are easily portable to other platforms that support them. OpenVMS VAX and OpenVMS Alpha provide support for such languages as Ada, C, COBOL, and Fortran.

2.1.3.3 OSF/Motif Applications

OSF/Motif, a component of the OSF Application Environment Specification (AES), is an industry standard for an open graphical user interface and its associated application programming interface (API). The OpenVMS graphical user interface is DECwindows Motif. Any Motif application can run on DECwindows Motif.

2.1.4 User Portability

User portability relates to the ease with which users can move between applications, experiencing a consistent user interface on different systems. Standard services and interfaces incorporated in open applications enhance user portability. In addition, user portability tools that are based on standards permit development of consistent user interfaces. Standards-based tools allow end users, developers, and managers to move easily from tool to tool, eliminating the need for retraining.

Standards-based user interface software available in the OpenVMS open systems environment includes OSF/Motif and the POSIX for OpenVMS shell and utilities.

2.1.4.1 OSF/Motif User Interface

The OSF/Motif interface implemented by DECwindows Motif provides users with a consistent screen appearance and behavior for applications that conform to the X Window System. (For a description of the DECwindows Motif graphical user interface, see Section 5.2.2.)

2.1.4.2 POSIX User Interface

The POSIX interface in the OpenVMS operating system allows users to develop and run applications that can also be run on any other platform that supports POSIX and X/Open standards and draft standards. POSIX 1003.2 defines the shell and utilities services that comprise a user interface environment similar to the Korn shell in the UNIX environment. Included in the interactive shell are commands similar to UNIX commands. OpenVMS users running POSIX for OpenVMS can experience the look and feel of UNIX. (For a description of the POSIX for OpenVMS user interface, see Section 5.3.)

2.1.5 Multivendor Interoperability

Interoperability is the ease with which systems from multiple vendors can work together, sharing data and integrating applications. Interoperability and integration of systems from different vendors involves connecting the systems. All systems and their resources are widely available in an open environment. In addition, an open system interoperates with any installed computing environment.

Multivendor interoperability provides the following benefits:

- Permits easy interconnection of different multivendor systems throughout the enterprise to communicate and share data and applications
- Provides the flexibility to change and expand the computing environment to meet changing needs
- Allows unimpeded access to systems and use of data

In an OpenVMS environment, applications can interoperate with different applications running on other systems that support the same standards. OpenVMS supports the capabilities supplied by networking software and protocols, middleware for integrating applications on different vendor systems, and DCE services for a distributed computing environment.

2.1.5.1 Open Networking Capability

OpenVMS networking software that provides open, multiprotocol, multivendor networking capability includes the DECnet/OSI for OpenVMS VAX and DECnet/OSI for OpenVMS Alpha products, based on ISO standards for the Open Systems Interconnection (OSI) model. The OSI model allows free interconnection of systems from different vendors in an open systems network. Participation in the network is open to any vendors that conform to the appropriate ISO standards. DECnet/OSI for OpenVMS allows users to choose between OSI protocols, DNA protocols, and TCP/IP protocols.

Digital TCP/IP Services for OpenVMS, an optional software product, supports connection to UNIX systems. The NFS server software on OpenVMS allows UNIX clients to access both the OpenVMS file system and files compatible with UNIX on the OpenVMS system. OpenVMS systems running TCP/IP products can serve as gateways to the Internet.

These networking products are summarized in Section 1.3.4. Distributed networking capabilities are discussed in Section 2.2.5, and the distributed heterogeneous networking environment is covered in Section 6.2.

2.1.5.2 Middleware Support for Open Systems

Middleware products are based on open international standards and specifications, including the AES and DCE standards developed by OSF. Middleware (software layered between the operating system and applications) provides application programming interfaces (APIs) to accomplish the dialogue between applications and their users, data, systems, and other applications.

Middleware is the Digital implementation of an open systems profile that provides presentation, communication, control, information, computation, and management services. Middleware provides for interoperability between applications in an open systems environment (see Section 2.2.3). Middleware products can be used to integrate applications across a distributed, multivendor environment (see Section 6.3).

2.1.5.3 DCE Services Support for Interoperable Applications

Multivendor interoperability capabilities are also provided by OSF DCE services being implemented by Digital. The OSF DCE consists of a set of basic mechanisms that allow application developers to design interoperable applications. (The DCE software environment is described in Section 2.2.4, and Digital DCE product functionality is covered in Section 6.3.4.)

2.2 Distributed Computing Capabilities

In a distributed computing environment, an application is distributed over two or more systems or processors, each of which has its own autonomous operating environment. A distributed application is composed of separate modules, running on different systems, that communicate with each other by passing data between modules or by sharing access to files or databases. A distributed application must be able to coordinate its activities over a dispersed operating environment. (Distributed and centralized styles of computing are contrasted in Section 1.2.2.)

The distributed computing environment can consist of software located in a single box or single room or can comprise a worldwide network of computers. The systems in the distributed configuration can be uniprocessor, multiprocessor, or VMScluster systems; systems from different vendors can be included in the same configuration.

2.2.1 Client/Server Style of Computing

One style of distributed computing that permits resource sharing between different systems is client/server computing. In the client/server environment, portions of an application are distributed across the network between servers and clients:

- A server is any system that provides a service or resource to other systems.
- The client is the system requesting the service.

This style of computing allows each portion of a distributed application to run in its own optimal environment. The whole application does not have to run on one centralized system (such as a mainframe system), but enterprisewide cohesiveness can still be maintained. For example, individuals or local offices, using their own computers and running software appropriate to their needs, can be linked to large computers or VMScluster systems in a network. A distributed

computing system can function as though it were a single system that connects all parts of an enterprise. The client can have transparent access to the integrated resources of the enterprise.

Any system can be a client or a server, and some systems may include both client software for certain applications and server software for other applications. Servers can be connected to many clients, and a client can be connected to more than one server at a time. (Client and server relationships may change frequently: at times it may not be possible to tell which is the client and which is the server.) In some cases, the application is stored on the server and run on the client, using the resources of the client. The user, who does not need to know what system is serving the application, can function in a familiar, local environment.

2.2.2 OpenVMS Client/Server Capabilities

OpenVMS systems support a wide variety of client/server configurations. Clients requiring resources can be personal computers, workstations, point-of-sale devices, OpenVMS systems, or systems from other vendors that are running the appropriate client software. Users on client systems can use character-cell terminals or windowing desktops.

Servers fulfilling clients' requests can be located on OpenVMS systems or other operating systems running appropriate server software. OpenVMS servers, for example, can provide file access, printing, application services, communication services, and computing power as application engines to clients on desktop devices or in laboratories or factories. Client/server configurations permit the commercial-strength capabilities of OpenVMS host systems to be integrated with the personal-computing capabilities of desktop systems.

Middleware, which runs on OpenVMS and other systems from multiple vendors, can be used to tie together clients and servers. Middleware integrates various client and server systems through application, communication, data interchange, and multivendor support (see Section 6.3). Complex information-sharing environments involving PC clients and operating system servers are supported by PATHWORKS software (see Section 2.2.2.2).

The following sections provide examples of different kinds of client/server relationships involving OpenVMS systems. Software in OpenVMS distributed production server environments is summarized in Section 2.3 and described in detail in Section 7.1. Software in client/server environments is described in Section 6.4.

2.2.2.1 OpenVMS Servers with OpenVMS Clients

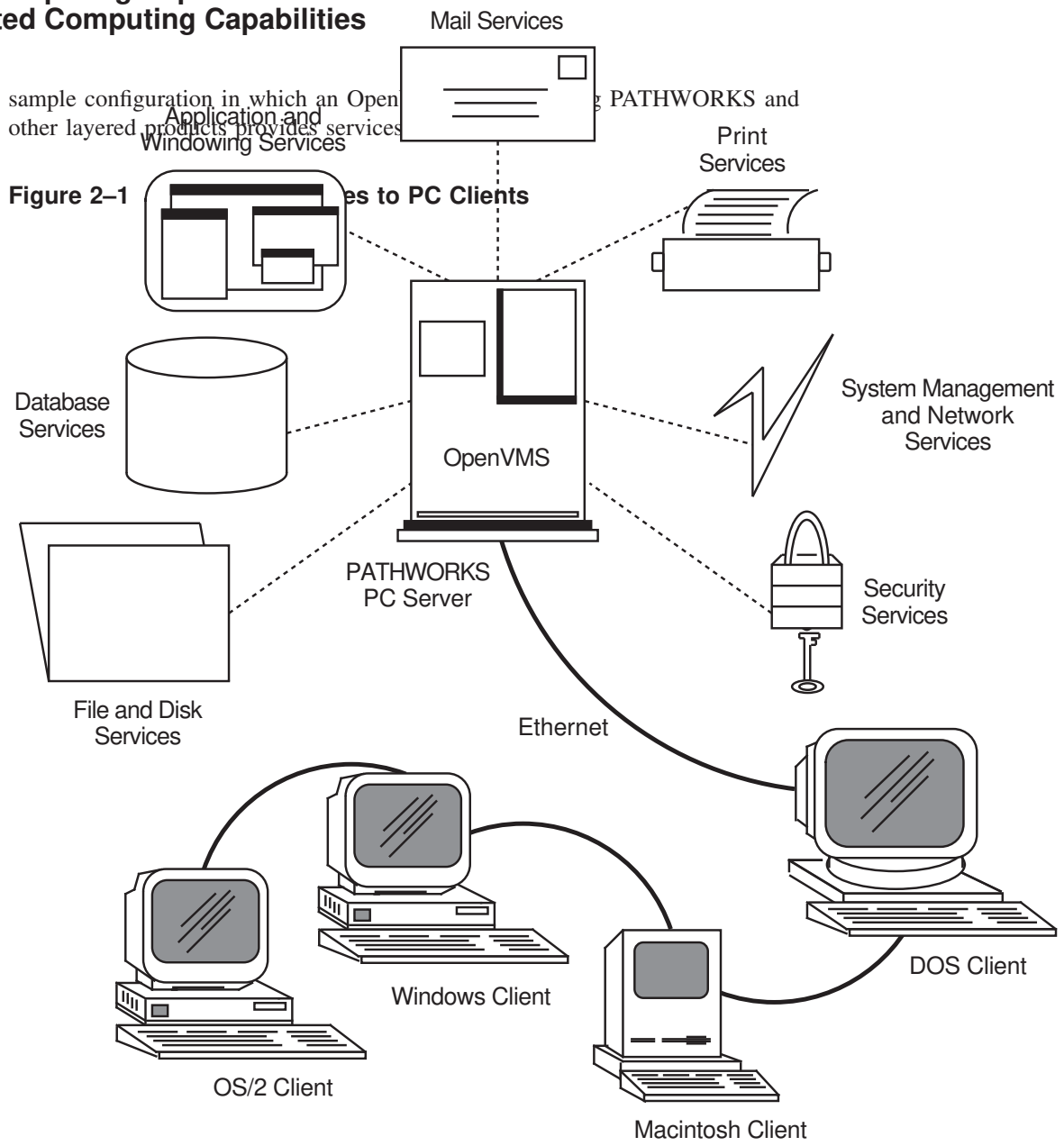
OpenVMS nodes in a VMScluster are clients of distributed services provided by the VMScluster system (as described in Section 3.4.1). A VMScluster system can act as a disk, tape, file, print, and batch server to cluster members. Client/server relationships on VMSclusters are described in Section 6.4.1.

2.2.2.2 OpenVMS Servers with PC Clients

The OpenVMS operating system running on a VAX or Alpha platform can act as an application, file, and print server for large groups of personal computer clients through DECnet or TCP/IP networking connections. Servers and clients can be connected using PATHWORKS products: PATHWORKS server software on the OpenVMS or VMScluster system and PATHWORKS client software supported on a variety of personal computers (including MS–DOS, Windows, Windows 95, Windows NT, OS/2, and Macintosh operating systems). Figure 2–1 shows a

sample configuration in which an Open g PATHWORKS and other layered products provides services

Figure 2–1 es to PC Clients



ZK–5462A–GE

PATHWORKS is the personal computing system architecture that integrates personal, departmental, and enterprisewide computing environments so that PC users can directly access shareable information and resources throughout the enterprise. PC and terminal users can share applications, data, and system resources such as printers, CD readers, and network gateways without losing the benefits of industry-standard personal computing.

The OpenVMS server can use PATHWORKS for OpenVMS, in conjunction with networking software, to provide file, print and electronic mail services, system management and network services, security services, and windowing services to PC clients. An OpenVMS server running additional layered products can supply database services and transaction-processing services to PC clients.

The use of OpenVMS servers in PATHWORKS environments is described in detail in Section 6.4.2.

2.2.3 Middleware in Distributed Environments

Middleware, based on open standards, provides services for distributed applications. The same services that run on OpenVMS can run on any distributed system that uses middleware products. Digital middleware products provide the basic computing foundation to enable applications to be more interoperable and distributable across different computer systems: for example, OpenVMS systems, Digital operating systems based on UNIX (such as Digital UNIX), and other vendor systems (such as UNIX, MS–DOS, OS/2, Macintosh, Sun, and IBM systems).

Using middleware, applications can easily move across different systems in a network and have the systems all work together as if they came from one vendor. With applications that employ middleware, users can more easily access and share information, no matter where they are located or what kind of system they are using.

2.2.4 Distributed Software That Is Compliant with OSF Standards

With the Distributed Computing Environment (DCE), the OSF has established a standard set of software services and interfaces that facilitate the creation, use, and maintenance of client/server applications. The Digital DCE product family provides a certified set of the distributed computing functionality specified for the OSF DCE, as well as tools for application developers.

Digital DCE serves as the basis for an open computing environment where networks of multivendor systems appear as a single system to the user. DCE makes underlying networks and operating systems transparent to application developers. Therefore, application developers can easily build portable, interoperable applications, and users can locate and share information safely and easily across the whole enterprise.

The optional Digital DCE for OpenVMS product family (described in Section 6.3.4) includes DCE Run-Time Services for OpenVMS and the DCE Application Development Kit for OpenVMS, which includes:

- DCE Remote Procedure Call (RPC)
- DCE Threads Service
- Documentation and sample applications for developing RPC applications.

Both the run-time services and the development kit are available on OpenVMS VAX and Alpha platforms.

Digital also supports the following DCE products:

- DCE Cell Directory Server (CDS), which is a location-independent name directory service
- DCE Security Server, which provides authentication and authorization services for DCE users
- Resource Broker, which permits allocation of resources to servers based on availability and efficiency

2.2.5 Distributed Networking Capabilities

OpenVMS networking support for distributed processing is provided by DECnet software and by the Digital TCP/IP Services for OpenVMS product (see Section 1.3.4). The use of networking software in multivendor distributed environments is discussed in Section 6.2.

The DECnet family of communication products (software and hardware) allows the OpenVMS operating system to participate in the DECnet network. DECnet for OpenVMS, the Phase IV product, supports the DNA protocols that permit communication with other systems supporting compatible versions of the DNA protocols. DECnet/OSI for OpenVMS, the Phase V product, supports OSI protocols that permit communication with other vendors' systems that support OSI, as well as DNA protocols, and provides links to permit DECnet or OSI applications to run over TCP/IP connections. Users of DECnet/OSI for OpenVMS can choose between OSI, DNA, and TCP/IP networking protocols, which can run simultaneously. Nodes running DECnet/OSI for OpenVMS and DECnet for OpenVMS can be configured to coexist on the same network.

DECnet/OSI networking supports time and name services for distributed processing over the network:

- The DECdts time service is a method for coordinating and setting system time over the network. DECdts supplies time-synchronization features essential to networks that are running distributed applications.
- The DECdns name service provides for networkwide naming of objects. DECdns supplies distributed applications with a consistent, networkwide set of names called a namespace. This namespace makes it possible for users and applications to refer to resources on the network (such as files, disks, and nodes) by using a single name, without having to know where the resource is located. The DECdns software uses a client/server model.

DECnet/OSI for OpenVMS supports the distributed file service (DECdfs) that provides users and applications with transparent, high-performance file access while using fewer CPU resources than standard DECnet file access. DECdfs systems can act as clients, servers, or both clients and servers.

Digital TCP/IP Services for OpenVMS, an optional product, provides for TCP/IP connections and supplies NFS server capabilities. These services permit OpenVMS VAX and OpenVMS Alpha to become full participants in TCP/IP networks. The Digital TCP/IP Services for OpenVMS include a set of industry-standard communication protocols (TCP, IP, FTP, Telnet, and other protocols), Digital Remote Procedure Call (DECrpc) for OpenVMS, and NFS server software. NFS software permits UNIX clients to access OpenVMS files and files compatible with UNIX stored on the OpenVMS system. (For additional information about the TCP/IP services supported, see Section 6.2.2.)

2.2.6 Distributed Features of DECwindows Motif

DECwindows Motif, based on industry-standard OSF/Motif, permits users to access application programs running on other machines in the network as if the applications were running locally. With the DECwindows software, multiple device-independent applications can run simultaneously in various separate workstation windows. Applications function as clients and the DECwindows program that responds to the applications is the DECwindows server. (The DECwindows Motif user interface is described in Section 5.2.2.)

2.3 High-Integrity Production System Capabilities

OpenVMS offers commercial-strength computing capabilities to meet the needs of production systems. A production system is a computing system or configuration critical to the operation of an organization. OpenVMS supplies the computing power and dependability features needed to keep a mission-critical production system available as required while ensuring the integrity of the data. OpenVMS systems and related optional software support predictable, stable computing environments.

The high-integrity, availability, manageability, and security features of OpenVMS VAX have been proven in applications tested and enhanced over the years. These features are now implemented on OpenVMS Alpha systems. OpenVMS VAX applications can migrate easily to OpenVMS Alpha platforms. Applications developed on OpenVMS VAX and Alpha systems, including applications developed by other vendors, cover the full range of activity from technical research to commercial data centers.

The following sections discuss OpenVMS production system capabilities, summarize the characteristics of a dependable OpenVMS system, and describe OpenVMS software availability tools and data integrity tools. They also summarize transaction-processing capabilities and manageability and security features for production systems. Additionally, OpenVMS supports database software supplied by other vendors to provide full production system capabilities. For a discussion of the growth potential of OpenVMS systems, see Section 1.5.

Software used in the OpenVMS distributed production server environment is described in Section 7.1.

2.3.1 Dependable OpenVMS Computing Systems

A dependable computing system is one that can be relied on to provide services to its users when the services are needed. A dependable system also provides sufficient performance so that users and applications can conduct their work efficiently.

The following types of OpenVMS systems meet different levels of dependability requirements for production systems:

- Conventional computing systems (for example, a standalone system or systems connected by a LAN)
- Highly available computing systems (for example, a VMScluster system using shadowed disks and transaction processing software): For business functions requiring computing services that are uninterrupted for essential time periods, with minimal down time
- Disaster-tolerant computing systems (for example, the Business Recovery Server consisting of linked data centers located at a safe distance from each other): For business operations that must remain uninterrupted and must be immune to any type of foreseeable disaster

The components in a dependable system have the following characteristics:

- Reliability: The ability to maintain a functioning condition, through the use of fault-prevention strategies
- Recoverability: The ability to return to a functioning condition from a nonfunctioning or incorrectly functioning condition, through the use of failure-recovery strategies

- Fault tolerance: The ability to exhibit an apparently functioning condition, through the use of error detection and correction strategies

2.3.2 Availability Tools

OpenVMS availability tools include VMScluster systems, volume-shadowing software, and tools such as an availability management software product:

- VMScluster systems are loosely coupled multiprocessor configurations that allow designers to configure redundant hardware that can survive equipment failures. The OpenVMS operating system, which runs on each processor node in a VMScluster system, provides a high level of transparent data sharing and independent failure characteristics. The processors interact to form a cooperating distributed operating system. All disks and their stored files are accessible from any processor as if those files were connected to a single processor. (See Section 1.3.3 for summary information on VMScluster configurations and Section 3.4.1 for a description of the VMScluster software.)
- Volume shadowing is a technique that provides data availability to computer systems by protecting against data loss from media deterioration, communication path failures, and controller or device failures. The process of volume shadowing entails maintaining multiple copies of the same data on two or more physical volumes. Up to three physical devices are bound together by the volume-shadowing software into a shadow set or virtual unit. Volume-shadowing software replicates data across the physical devices. Applications access the virtual unit as if it were a single standard, physical disk. If one volume of the shadow set fails, requests for data are automatically directed to another volume, allowing operations to continue without interruption.

  Phase II Volume Shadowing, offered for both OpenVMS VAX and OpenVMS Alpha systems, is a fully distributed, clusterwide data availability product intended to service Digital and SCSI storage architectures. All essential shadowing functions are performed within the OpenVMS operating system (see Section 3.4.2).
- DECamds (the Digital availability manager for distributed systems) monitors, investigates, diagnoses, and corrects OpenVMS system resource utilization. DECamds permits the system manager to identify and resolve areas of resource denial on every system on the network. The DECamds console runs on an OpenVMS VAX or OpenVMS Alpha system, and related drivers run on each VAX and Alpha system on the network. (DECamds is described in Section 7.1.2.1.)

2.3.3 Data Integrity Tools

Data integrity tools available on OpenVMS VAX and Alpha systems include RMS Journaling software and DECdtm software:

- RMS Journaling software helps protect the data integrity of Record Management Services (RMS) files on OpenVMS systems by recording file updates in a separate journal file. The journal file is used to restore the file to its original state if a failure should occur. Recovery-unit journaling ensures RMS operations on a file are either all completed or not done at all, to protect transaction integrity. (See Section 3.4.3 for a description of RMS Journaling software.)

- The DECdtm distributed transaction manager is a set of services to facilitate transaction processing. These services enable the application designer to implement atomic transactions, using an optimized two-phase commit protocol. DECdtm provides the enabling technology and features for distributed transaction processing, ensuring both transaction and database integrity across multiple resource managers (see Section 3.1.6).

2.3.4 Transaction-Processing Capabilities

OpenVMS VAX and OpenVMS Alpha support optional software products that provide essential production system capabilities, as follows:

- Transaction-processing monitor: ACMS, the Application Control and Management System, is the highly reliable transaction-processing monitor that works with Digital and other vendor commercial software to provide development and run-time environments for transaction-processing applications. ACMS provides dependability features, including application failover, to keep applications running reliably. ACMS is suitable for very large online transaction-processing (OLTP) applications for mission-critical business use. ACMS is described in Section 7.2.2.
- The Digital forms management package, DECforms, integrates with ACMS to provide forms-processing capabilities in transaction-processing environments. DECforms combines the traditional capabilities of previous Digital forms systems and adds new features (see Section 5.4.1).

In addition, the DEC Reliable Transaction Router (RTR) is a distributed, fault-tolerant software message routing system that supports the implementation of reliable and distributed transaction-processing systems. DEC RTR is a high-availability application used in several very large production systems around the world (for example, in stock/option markets). DEC RTR is described in Section 7.2.4.

A transaction may span multiple nodes of a cluster or network. Support provided by DECdtm services allows multiple resource managers, such as OpenVMS RMS software and other vendor database products, to be combined in a single transaction. Applications can define distributed transactions that may include calls to any of the data management products. The distributed transaction will either commit or abort as a single transaction.

The OpenVMS system facilitates database sharing among applications that perform transaction processing and those that do not and sharing with decision-support systems and remote nodes requesting data.

2.3.5 Manageability and Security for Data Centers

OpenVMS provides a wide range of tools and utilities to help users manage complex computer environments, such as a large data center. Basic OpenVMS system management capabilities are integrated in the OpenVMS operating system and are augmented by optional products, such as the POLYCENTER Console Manager, which serves as a control center for managing all connected members of the cluster. Overall OpenVMS system and network management capabilities are described in Section 2.4, and optional management products running on an OpenVMS production server are summarized in Section 7.1.3.4.

The software facilities and tools for managing complex systems include the following:

- The optional Business Recovery Server is designed to permit management of disaster-tolerant configurations. The Business Recovery Server is

an integrated software package that permits VAX and Alpha CPUs in multiple data centers to be combined into a single functional, security, and management domain. The server integrates clustering, volume shadowing, and related management and interconnect technologies into a manageable VMScluster system, consisting of data centers at widely separated locations connected by FDDI, or by FDDI and ATM or T3 communications services. The Business Recovery Server can be managed from any location, even if a total data center failure occurs.

The Business Recovery Server is discussed further in Section 7.1.3.5.

- Storage management software developed by Digital provides software tools for managing mass storage and the data objects (files) stored on it in a distributed environment of heterogeneous computing systems. Examples of optional storage management products that run on OpenVMS VAX and OpenVMS Alpha are:
  - StorageWorks RAID Software for OpenVMS: Manages groups of disk drivers as arrays in a VMScluster
  - POLYCENTER File Optimizer for OpenVMS: Provides for file defragmentation
  - Storage Library System (SLS): Automates VMScluster backup operations
  - POLYCENTER Hierarchical Storage Management (HSM) for OpenVMS: Performs file-shelving operations
  - Disk Striping Driver: Combines multiple disks into a single striped virtual disk (called a stripe set)

  These storage management tools are described in Section 7.1.3.1.
- OpenVMS also supports sophisticated tools for monitoring and tuning performance. The capabilities of existing performance management and capability-planning products have been incorporated into an integrated product set called POLYCENTER Performance Solution (as described in Section 7.1.3.3).
- The DECram for OpenVMS device driver, which creates a pseudodisk in main memory, can be used to improve I/O performance.
- The integrated security controls designed into the OpenVMS operating system (described in Section 3.3), combined with the capabilities offered by optional products that run on OpenVMS, guard data integrity and availability. In a local area network environment, optional encryption and decryption services provide protection for data being passed between systems.

2.4 System and Network Management Capabilities

OpenVMS reduces the complexity of managing individual systems, clusters of systems, and entire multivendor computing environments through the use of its own built-in management capabilities and optional software that runs on OpenVMS.

2.4.1 Managing OpenVMS Systems

OpenVMS provides, as an integral part of the operating system, system management capabilities that minimize the requirement for staffing of the system management function. Basic OpenVMS management capabilities include performance monitoring, tuning, security, storage management, and queue management (see Section 3.2). OpenVMS also supports a large number of applications and services provided by Digital and independent software developers for managing computer operations, including multivendor configurations.

OpenVMS system management tasks are ordinarily performed by a system manager. On a standalone workstation, the OpenVMS user usually serves as the manager of the OpenVMS system. In large installations, more than one system manager may be involved and some tasks may be performed by a computer operator.

OpenVMS systems to be managed range from standalone workstations to large VMScluster systems composed of high-end Alpha and VAX computers. Managing a VMScluster system is similar to managing a single system. In general, the tools used and the security and performance considerations are the same.

The OpenVMS Management Station is a powerful, easy-to-use, Windows based management tool that runs on a PC and performs account management across multiple systems. For a description of the OpenVMS Management Station, see Section 3.2.4.

Some integrated management utilities are applicable to all OpenVMS systems and clusters. Examples are the Backup, Monitor, and Mount utilities. The System Management utility (SYSMAN) is used to perform cluster management tasks, such as setting up a clusterwide environment and executing commands on a cluster. For a description of OpenVMS system management software, see Section 3.2.

Optional software applications supply system management functions for complex VMScluster environments. Examples are the POLYCENTER Scheduler and the POLYCENTER Console System. The optional Business Recovery Server manages large disaster-tolerant configurations.

OpenVMS can manage other systems remotely over the network. The POLYCENTER Software Distribution Manager, formerly known as the Remote System Manager, which runs on OpenVMS, is an optional networking product that permits a system manager to manage a number of OpenVMS systems and UNIX based systems connected to a DECnet network.

Using an OpenVMS management server can simplify management of remote systems and software. In a complex networking environment, a management server can provide effective control of distributed resources.

2.4.2 Managing Networks

Digital integrates network management operations into the network itself to keep applications and enterprises running despite changes in the network. The network automatically collects and stores data about itself, reports on network events, and takes action based on this information. Users can access this information and make online adjustments from anywhere in the network using network management software that exists on each node in the network.

DECnet/OSI provides modular, distributed network management capabilities, permitting remote management of all network functions. The command line interface to DECnet/OSI network management is the Network Command Language (NCL), which is used to manage all DECnet/OSI nodes and their entities.

DECnet Phase IV nodes use Network Control Program (NCP) commands to interface with network management. DECnet Phase IV network management functions include configuring, monitoring, and testing DECnet and performing host services to remote nodes (such as downline loading and upline dumping).

DECnet/OSI and DECnet Phase IV are compatible. DECnet/OSI nodes can invoke NCP to manage existing Phase IV nodes on the same network.

For additional information about DECnet network management in an open, distributed environment, see Section 6.2.

2.4.3 Managing Integrated Enterprises

OpenVMS supports the management of complex, distributed, multivendor computing environments. Digital POLYCENTER management products provide for integrated management of networks, systems, applications, and databases from many sources. POLYCENTER applications are designed to interoperate with applications and frameworks that comply with OSF standards. POLYCENTER products provide an open, extensible management platform for integrating network and system management applications.

The use of Digital software to manage enterprisewide multivendor environments is described in Section 6.3.6.

Part II

OpenVMS Systems Software

This part of the manual provides a summary overview of the software capabilities of the OpenVMS system and related optional software products, as used in an OpenVMS standalone or VMScluster configuration.

Chapter 3 describes the base OpenVMS operating system components and optional software integrated in the operating system. Chapter 4 summarizes programming software integrated in the system that supports application program development. Chapter 5 describes user interfaces and general user software provided with the OpenVMS system.

As described in this part of the manual, standalone OpenVMS systems and OpenVMS systems in a VMScluster configuration can supply full capabilities for interactive, timesharing, and real-time processing. For a description of the use of OpenVMS and VMScluster software in distributed, multivendor computing environments, see Part III.

3
Description of OpenVMS System Software

The OpenVMS operating system is the group of software programs that control computing operations and coordinate and manage the resources of the processing system. The OpenVMS Alpha operating system software runs on an Alpha processor, and the OpenVMS VAX system software runs on a VAX processor. The operating system software is loaded into memory when the system is initialized (booted) and remains there until the system is shut down. It performs both automatically and manually and supplies the operating environment for the user.

This chapter describes the software components of the base OpenVMS operating system, including system management and security software. It also describes optional software integrated into the OpenVMS system, including VMSclusters, volume shadowing, and RMS Journaling software.

3.1 OpenVMS Operating System Components

The OpenVMS operating system is made up of thousands of program modules that are grouped into layers (as shown in Figure 3–1):

- OpenVMS operating system kernel, supported by system services
- OpenVMS core services
- OpenVMS utility programs

The following sections describe these layers in detail.

In Figure 3–1, the privileges required for each layer are indicated by the corresponding access mode. Access modes range from the innermost (the kernel) with the most privilege, to the outermost (the user) with less privilege. The executive includes the parts of the operating system that are loaded into and executed from system space.

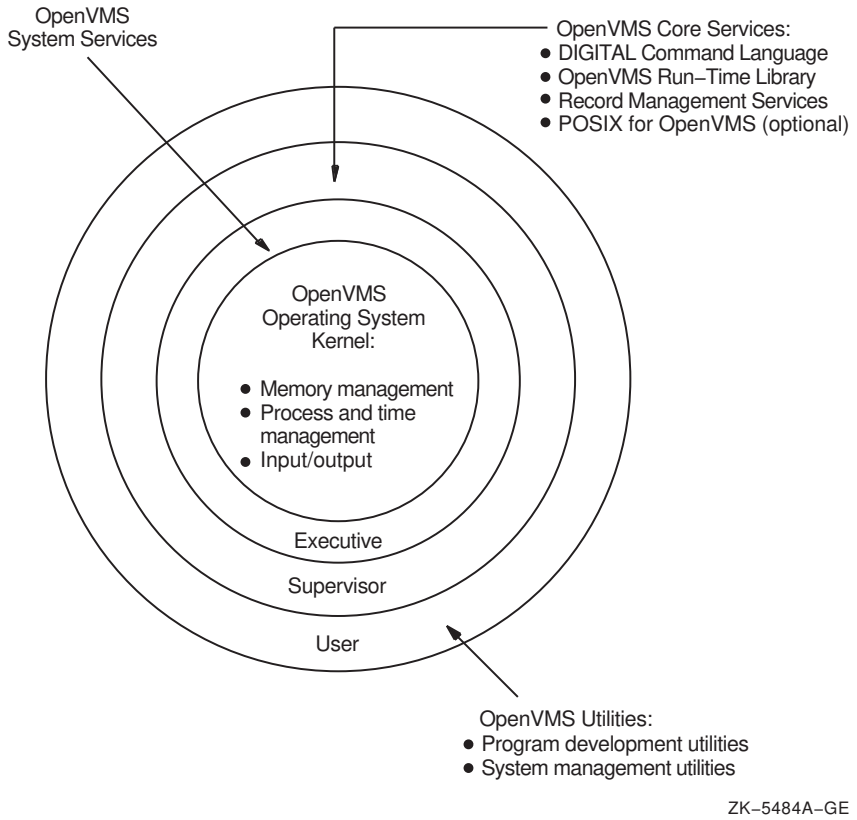
In the OpenVMS system, each program (known as an image) executes in the context of a process, as scheduled by the OpenVMS system software. A process consists of a process address space and hardware and software context. Suitable process privileges are assigned when a process is created.

3.1.1 OpenVMS Kernel

The OpenVMS kernel includes the programs, data structures, and related system services essential to the operating system. The programs in the kernel perform and coordinate processor operations and provide a multiuser, multitasking operating environment. The kernel programs are composed of three subsystems:

- Memory management
- Process and time management

Figure 3–1 OpenVMS Operating System Components



- Input/output (I/O)

Systemwide, protected data structures that support the kernel software include pager data structures, the I/O database, and scheduling queues.

3.1.1.1 Memory Management Subsystem

The memory management subsystem controls the allocation of memory resources and implements the OpenVMS system's virtual memory operation. Virtual memory management permits the development and use of programs larger than physical memory and allows any number of jobs to share memory.

To optimize the use of physical memory, the memory management subsystem causes large segments of code and data to be moved in and out of physical memory as they are needed. This process is carried out by the swapper mechanism. The basic set of memory locations used in memory mapping is called the page. (The value of the page is CPU specific.) The paging mechanism brings pages of an executing process into physical memory when referenced. When a process is executing, all of its pages are said to reside in virtual memory.

Both OpenVMS VAX and OpenVMS Alpha systems support 32-bit virtual addressing. OpenVMS Alpha also supports 64-bit virtual addressing, which makes the 64-bit virtual address space defined by the Alpha architecture available to both the OpenVMS Alpha operating system and its users. The 64-bit addressing capability allows per-process virtual addressing for accessing dynamically mapped data beyond 32-bit limits, and provides support for very large capacity physical memory.

The 64-bit space may be allocated or accessed using new system services or language features. 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.

Nonprivileged programs may optionally be modified to exploit 64-bit support.

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

3.1.1.2 Process and Time Management Subsystem

Process and time management functions include scheduling jobs for processing and performing process control. OpenVMS schedules processor time and memory residency on a priority basis. All processor operations are assigned a priority and executed based on that priority. When multiple requests are waiting at the same time, the scheduler starts and stops jobs on the basis of their priority. When an event (such as an I/O interrupt) occurs, the system processes the event and passes control to the highest priority process ready to execute.

On an OpenVMS Alpha system, process scheduling capabilities can be extended through the use of kernel threads, which allow a process to run concurrently on multiple CPUs in a multiprocessor system. A thread is the sequential flow of execution within a process's address space. A single process contains an address space in which a single thread or multiple threads execute concurrently. Programs typically have a single flow of execution and therefore a single thread. A multithreaded program has multiple points of execution at any one time. A multithreaded process can execute code flows independently in more than one CPU at a time. The threaded application can make better use of multiple CPUs in a symmetric multiprocessing (SMP) system.

The interface to kernel threads is through DECthreads run-time routines (see Section 4.3.1), which are used to schedule individual user mode application threads. A callback mechanism between the OpenVMS scheduler and the thread scheduler of DECthreads permits reduction of inherent scheduling latencies and a resulting increase in application speed.

3.1.1.3 I/O Subsystem

The OpenVMS I/O subsystem consists of OpenVMS device drivers and their associated data structures, as well as related system services. The I/O subsystem is responsible for processing I/O requests received from other layers of the OpenVMS operating system or from application programs. Device drivers initiate, manage, and respond to device interrupts to transfer data to and from the device. Each type of I/O device requires its own driver. Digital supplies drivers for all devices supported by the OpenVMS operating system and provides $QIO service routines to access the special device-dependent features available in many of these devices.

Users can also write their own device drivers. OpenVMS VAX supports device drivers written in MACRO for devices not supplied by OpenVMS VAX. OpenVMS Alpha supports user-written device drivers written in higher level languages, including C and BLISS, as well as in MACRO-32. For information on writing device drivers in the C programming language, refer to *OpenVMS Alpha Device Support: Developer's Guide*. For reference material applicable to device drivers written in C and MACRO-32, see *OpenVMS Alpha Device Support: Reference*.

The OpenVMS Alpha System-Code Debugger can be used to debug system code and device drivers written in C, BLISS, or MACRO.

3.1.1.4 Supporting System Services and Facilities

The OpenVMS system services support and complement the kernel subsystems. The system services control system resources available to user applications, provide communication and synchronization among processes, and coordinate I/O operations. User-written application programs can call system services directly (see Section 4.3.3).

OpenVMS provides the following interprocess communication facilities for applications that consist of multiple cooperating processes:

- Shared memory sections that permit multiple processes to have concurrent access to shared address space
- Common event flags that provide simple synchronization
- Mailbox virtual devices that allow processes to communicate with queued messages
- The lock manager, which provides a facility for synchronizing access to resources by allowing locking and unlocking of resources by name (see Section 3.4.1)

POSIX for OpenVMS supports real-time functions for interprocess communication between multiple processes in the area of event notification, message queues, and shared memory (see Section 4.4). POSIX applications developed in the POSIX for OpenVMS environment can call on OpenVMS services directly. However, use of system services that are specific to the OpenVMS operating system will affect the portability of the POSIX application.

3.1.2 OpenVMS Core Services

OpenVMS core services support basic user and application interaction with the processing system. The core services include the following:

- DIGITAL Command Language (DCL): Command-line interface to the OpenVMS operating system
- OpenVMS Run-Time Library: A library of run-time routines for performing a variety of commonly required functions
- OpenVMS Record Management Services (RMS): A device-independent method of handling I/O
- Optionally installable POSIX for OpenVMS: Portable operating system interfaces that conform to IEEE standards and enable users to write applications that can be moved from one system conforming to POSIX and X/Open standards to another and used on both of them

DCL is a traditional user interface to OpenVMS that supports batch or interactive operations. DCL provides commands for requesting system actions from manipulating files to running applications (see Section 5.2.1).

The OpenVMS Run-Time Library provides commonly required application functions and other general-purpose functions. These routines can be called by programs written in all the programming languages supported by OpenVMS. All routines in the run-time library follow the OpenVMS calling standard and most are contained within a shareable image. (OpenVMS Run-Time Library routines are described in Section 4.3.2.)

The OpenVMS RMS facility provides a set of general routines for file and record operations on OpenVMS systems. RMS is a high-level interface to the file system and OpenVMS I/O subsystem. It is used by most optional (layered) products for file and record operations and is the default I/O service for all programming languages that run on OpenVMS Alpha and OpenVMS VAX systems. RMS routines can be used by application programs for creating files, opening and closing files, and reading, writing, and deleting records in files. (RMS is described in Section 4.8.)

POSIX for OpenVMS (described in Section 2.1.3.1) is optional software that, when installed, is tightly integrated into the OpenVMS environment and can take advantage of OpenVMS features. However, if a POSIX for OpenVMS application uses features available in the OpenVMS environment that do not conform to POSIX and X/Open standards, the portability of the POSIX application is affected. (For example, using the POSIX dcl utility, which allows access to OpenVMS services within the POSIX session, affects the application's portability.) OpenVMS technologies, such as clustering and volume shadowing, are available transparently to POSIX applications. (POSIX for OpenVMS programming is described in Section 4.4, the user interface in Section 5.3, and POSIX files in Section 5.5.2.)

3.1.3 OpenVMS Utility Programs

The OpenVMS utility programs include system management and user utilities and program development facilities. See Table 3–1 for examples of OpenVMS utility programs.

Table 3–1 Examples of OpenVMS System Utility Programs

Type of Utility	Examples of OpenVMS Utility Programs
System management utilities	AUTHORIZE, AUTOGEN, BACKUP, MONITOR, MOUNT, SYSMAN
User utilities	MAIL, SORT/MERGE, Help
Text-processing utilities	Extensible Versatile Editor (EVE), EDT editor
Program development utilities	Linker, debugger, LIBRARIAN, FDL

OpenVMS provides system managers and users with many powerful utilities to control the day-by-day operations of their systems. OpenVMS system management utilities are described in Section 3.2. OpenVMS users can access utilities that assist them in performing daily operations and communicating with other users. User utilities provided with the OpenVMS operating system, including editors and text processors, are discussed in Chapter 5. OpenVMS programming utilities include text processors for creating source programs and program development utilities for processing the programs. These programming utilities are described in Chapter 4.

Other optional software that runs in OpenVMS environments is described in Chapter 6 and Chapter 7.

3.1.4 OpenVMS VAX Vector-Processing Capability

The OpenVMS VAX operating system provides fully shared, multiprogramming support for VAX vector-processing systems. A vector is a group of related scalar values or elements (such as an array of numbers). Several midrange and high-end VAX processors have adopted an optional design for integrated vector processing using vector registers and vector instructions. A vector processor can routinely process a vector four or five times faster than a traditional processor. Vectors are useful for CPU-intensive applications involving repeated operations on groups of simple elements (for example, in weather forecasting, molecular modeling, and financial modeling).

OpenVMS VAX support for vector processing includes these features:

- Fast-vector math routines for high performance
- A standard OpenVMS VAX facility to permit writing and debugging of applications that use vectors on an OpenVMS VAX system that emulates the vector-processing environment
- Vector-processing operations supported by system management and the OpenVMS Accounting and Error Log utilities

Alpha processors do not offer the vector-processing option because their basic design provides for high-speed calculations.

3.1.5 OpenVMS Symmetric Multiprocessing

OpenVMS supports symmetric multiprocessing (SMP), which permits multiple processors to perform operations simultaneously, dividing and sharing the work load. SMP multiprocessors are tightly coupled and appear to the system and the user as a single system (see Section 1.2.2). OpenVMS SMP is available on selected VAX and Alpha processing systems.

Multiple processors execute code from a single shared memory address space, and users and processes share a single copy of OpenVMS. SMP also provides simultaneous shared access to common data in global sections available to all processors. SMP dynamically balances the execution of all processes across all processors, based on process priority.

DCL and system services interfaces provide access to the scheduling mechanisms that control processor affinity (the CPU on which the process is to run) and processor capabilities (resources required of the CPU).

The primary advantage of SMP is increased throughput. SMP can be optimized in two ways:

- Multiple processes can be executed simultaneously on different processors, maximizing overall system performance.
- Single-stream application programs can be partitioned into multistream jobs, minimizing the processing time for a particular program.

3.1.6 Digital Distributed Transaction Management Services

An individual transaction is a series of operations on a file or a set of files. In a distributed transaction, data can be distributed among different systems, increasing the difficulty of maintaining the integrity of the transaction-processing data.

The OpenVMS operating system supplies DECdtm services to support distributed transactions. DECdtm services provide for complete and consistent execution of distributed applications.

DECdtm services comprise a transaction manager, interfaces to system services, communication services, and logging and recovery services. The transaction manager supports application-supplied services to start, end, or abort transactions, and sends instructions to resource managers on how to complete the transaction.

DECdtm integrated services for distributed transaction management ensure both transaction and database integrity across multiple resource managers such as other vendor database systems, and across networks around the world. To ensure consistency of data, DECdtm services cause updates to all databases to occur as a single ''all or nothing'' unit of work, regardless of where the data resides physically.

DECdtm employs a two-phase commit protocol that enables application developers to build dependable distributed applications. The two-phase commit protocol is a handshaking procedure in which all participants in a distributed transaction first agree to commit and then, on a signal from coordinating software on the node that initiated the transaction, actually do commit. The commitment protocol guarantees that all parts of a transaction complete or the transaction has no effect on any involved resource manager (see Section 7.2.3).

DECdtm system services are used by an array of products that run on OpenVMS Alpha and OpenVMS VAX systems, including transaction-processing monitors, database products, and RMS Journaling. Transaction-processing products are described in Chapter 7.

3.2 OpenVMS System Management Software

The system management environment for OpenVMS Alpha and OpenVMS VAX systems is generally the same: most system management utilities, command formats, and tasks are identical on the two platforms. This section indicates any system-specific software.

OpenVMS system management tasks ordinarily include the following:

- Installing, upgrading, and updating software
- Starting up and shutting down the system
- Customizing the system
- Managing user access to the system and controlling system resources
- Managing and monitoring system security
- Managing peripheral devices and storage media, including public disks
- Managing system files and directories
- Backing up and restoring files
- Monitoring the system
- Tracking and reporting resource usage
- Maintaining acceptable system performance and tuning the system
- Managing batch and print queues

System managers may also be called upon to perform some of these tasks:

- Setting up and maintaining a network
- Setting up a VMScluster environment
- Managing special system configurations or processing environments

3.2.1 OpenVMS Installation and Configuration

The POLYCENTER Software Installation utility is used to install OpenVMS Alpha operating system software and can be used to install optional (layered) products on OpenVMS Alpha and OpenVMS VAX operating systems. The POLYCENTER Software Installation utility implements a new technology that simplifies the distribution and management of software. The utility provides DCL and DECwindows Motif interfaces that system managers can use to quickly configure, install, and upgrade the OpenVMS Alpha operating system and optional products. It also allows system managers to track the status of software on their systems. Starting with OpenVMS Alpha Version 6.2, it is possible to install or upgrade a target system disk from a running OpenVMS operating system.

For information about using the POLYCENTER Software Installation utility to install or upgrade the OpenVMS Alpha operating system, see the most recent OpenVMS Alpha Upgrade and Installation Manual. For more information about the utility, see the *POLYCENTER Software Installation Utility User's Guide*.

Any optional software that is not compatible with the POLYCENTER Software Installation utility can be installed on OpenVMS VAX or OpenVMS Alpha using the VMSINSTAL utility.

The installation software for the OpenVMS VAX operating system is an interactive VMSINSTAL command procedure that guides the system manager through each step of installing, updating, and upgrading software to a higher version number. During the OpenVMS installation procedure, VMSINSTAL sets up initial system parameters and installed images using AUTOGEN software and automatically brings up the system.

For information about installing OpenVMS VAX systems using VMSINSTAL, refer to the most recent OpenVMS VAX Upgrade and Installation Manual. VMSINSTAL is described in the *OpenVMS Developer's Guide to VMSINSTAL*.

Starting up a newly installed OpenVMS system involves booting the system, using either a nonstop or conversational boot. Nonstop booting is faster and easier because Digital sets typical system parameters for the hardware configuration. Conversational booting permits the system manager to change System Generation (SYSGEN) utility parameters during the boot procedure. On a standalone OpenVMS VAX system, to reduce startup time, a system snapshot can be taken and used for booting the system.

OpenVMS will boot on the VAX or Alpha processor, automatically configuring itself to the CPU, memory, and devices present. After a new configuration is booted, the system manager can use the AUTOGEN command procedure with its feedback mechanism to optimize system parameter settings.

When the system starts up, the system manager can log in and set up the environment. Each time the system starts up, it executes a series of startup command procedures, some of which the system manager can edit to customize the user environment. The startup command procedure invokes the appropriate utility to poll all connected I/O buses for devices and then configures itself to

allow access to those devices. The utility invoked to configure I/O on OpenVMS Alpha is the System Management utility; on OpenVMS VAX, it is the System Generation utility.

Table 3–2 gives examples of some OpenVMS utilities and DCL commands that pertain to system configuration tasks.

Table 3–2 Examples of OpenVMS Configuration Utilities and Commands

System Management Task	Utility or DCL Command
Automatically set system parameters by detecting devices installed in a configuration	AUTOGEN command procedure
Create an on-media file system	INITIALIZE command
Make a volume known to the system	MOUNT command
Manipulate device characteristics	SET DEVICE command
Start or stop CPUs in an SMP kernel	SET CPU command
Manipulate magnetic tape characteristics	SET MAGTAPE command
Manipulate printer characteristics	SET PRINTER command
Manipulate terminal characteristics	SET TERMINAL command
Examine and modify system parameters on active systems	System Generation utility
Configure the OpenVMS VAX I/O subsystem, loading and connecting device drivers to the system	System Generation utility
Configure the OpenVMS Alpha I/O subsystem, loading and connecting device drivers to the system	System Management utility
Define the system management environment across nodes in a VMScluster; enable device and processor control commands to take effect across a cluster	System Management utility
Manage disk space quotas	SYSMAN DISKQUOTA command
Perform installation and upgrade of OpenVMS Alpha software and optional software	POLYCENTER Software Installation utility
Automate the installation of software and software updates to OpenVMS VAX and optional software	VMSINSTAL command procedure

3.2.2 Management Software for the General OpenVMS Environment

In addition to configuring the OpenVMS system environment, the OpenVMS system manager performs a variety of tasks to manage the environment. Examples of OpenVMS utilities used to perform general system management tasks are given in Table 3–3.

Table 3–3 Examples of OpenVMS System Management Utilities

System Management Task	Utility
Report on system use and account for use of system resources	Accounting utility
Control access to the system and its resources; use commands to set up user accounts (SYSUAF.DAT)	Authorize utility
Perform backup of full volume or incremental files on mounted disks to tape or another disk and restore them	Backup utility
Selectively report the contents of the one or more event log files	DECevent utility†
Selectively report the contents of the error log file containing system error messages	Error Log utility
Set up an OpenVMS system to be a service node on a LAT configuration	Local Area Transport Control Program (LATCP)
Manage multiple software licenses	License Management Facility (LMF)
Monitor system activity	Monitor utility
Make the contents of a tape or disk available to the system	Mount utility

†Alpha specific

3.2.2.1 Controlling System Access

The system manager is responsible for managing user access to the system.

The Authorize utility permits the system manager to establish the individual controls that customize user accounts. When a user logs in to OpenVMS, the system uses the information in the user authorization file (UAF) to validate the login attempt, establish the account environment, and create a process with the specified attributes. The system manager can use the Authorize utility to add to or modify user records in the UAF and other security-related system databases. Section 3.3 describes security management in the OpenVMS system environment.

Some of the information in the UAF account record is as follows:

- User name and password for the account (the password is encrypted)
- User identification code (UIC), which identifies the user as a member of a group that can share data
- The disk device and directory containing files owned by the account
- Limits and quotas on reusable system resources
- System privileges that define processing activities the account's process can engage in
- Any limitation on allowed login times
- Qualifiers such as ''captive'' or ''restricted'' that limit access to the system

The OpenVMS Management Station is a new management tool that permits the OpenVMS system manager to manage user accounts on multiple OpenVMS systems (as in a VMScluster). The OpenVMS Management Station is installed on a Windows based PC (see the description in Section 3.2.4).

3.2.2.2 Managing Devices and Storage Media

Management of the OpenVMS environment includes managing peripheral devices and storage media. Peripheral devices include storage devices (such as disks, compact discs, and tapes) and I/O devices (such as terminals, terminal servers, printers, modems, and card readers). System managers or operators can use DCL commands to add devices, set device software characteristics, display device information, and set protection for the devices.

Storage management is increasingly important as the amount of data to be stored grows. Very large configurations supporting thousands of users generate a huge volume of data. The Digital approach to storage management is based on the foundation of OpenVMS computing, and includes optional network-based storage management products.

OpenVMS provides the tools to allocate, initialize, and mount storage devices, create disk volume sets, and set protection on volumes. The system manager can create logical names for files and directories and can plan and create public directories. The DCL command ANALYZE/DISK_STRUCTURE can be used to check and repair disk structure. Defragmentation of disks can be performed using the OpenVMS Backup utility or the optional POLYCENTER File Optimizer for OpenVMS. (Optional storage management products developed by Digital are described in Section 7.1.3.1.)

3.2.2.3 Backing Up the System

One of the system manager's responsibilities is to prevent the loss or destruction of data due to equipment failure or accidental deletion or corruption of files. The system manager or operator can use the OpenVMS Backup utility to make backup copies of volumes on magnetic tape, magnetic disk, or certain optical disks. The Backup utility provides full volume and incremental file backup for file-structured, mounted volumes and volume sets. Individual files, selected directory structures, or all files on a volume set can be backed up and restored. The Backup utility can also be used to restore a save set (a special file created by the Backup utility) or list the contents of a save set. The system manager or operator can use a screen-oriented interface to the Backup utility as well as a command-line interface.

A system manager who has access to the OpenVMS Alpha or OpenVMS VAX distribution compact disc can back up and restore the system using the menu-driven procedure included on the OpenVMS distribution compact disc. On OpenVMS Alpha systems, the system manager can also upgrade or install the operating system using this menu. For more information about using the new menu-driven procedure for backup operations, see the *OpenVMS System Manager's Manual*. For more information about installing and upgrading the OpenVMS operating system, see the most recent version of the upgrade and installation manual.

A system manager who does not have access to the OpenVMS Alpha or OpenVMS VAX operating system distribution compact disc can use standalone BACKUP to back up and restore the system disk. For more information about standalone BACKUP, see the *OpenVMS System Manager's Manual*.

The system manager can also assign a protection mask to files, devices, and other objects to prevent unauthorized access (see Section 3.3.2).

3.2.2.4 Monitoring, Maintaining, and Tuning the System

The system manager is responsible for ensuring that the system performs consistently at an acceptable level. OpenVMS provides software tools that help the manager to monitor the condition of the system and to maintain and tune the system.

Monitoring the system involves obtaining information about the system, its users, and the processes running on the system. Examples of software used to monitor the system include the following:

- DCL SHOW commands display information about system, device, and process conditions (providing options such as CLUSTER, DEVICES, ERROR, MEMORY, STATUS, SYSTEM). DCL command procedures can be developed to automate monitoring of system information.
- The Monitor utility is used to monitor different classes of systemwide performance data and to display it or save it in a file for later use. Examples of data that can be monitored at specified intervals include process activity, I/O activity, memory management activity, and two-phase commit transaction activity (as supported by DECdtm services).
- The Accounting utility is used to monitor resource usage. The utility processes system accounting files to produce reports and summaries that indicate the ways the system is used, how it performs, and, in some cases, how particular users use the system.
- Log files that report on system activity include the following:
  - Operator log file: The operator can access this file directly.
  - Error Log file: The system manager or operator can use the ANALYZE/ERROR_LOG command to analyze and view data in this file.
  - Security audit log file: The manager can use the Audit Analysis utility to analyze this file.

Maintaining acceptable system performance involves using the Monitor utility to develop a database of performance information, and using the Accounting utility to generate performance reports. With this information, the system manager can manage the work load on the system. For example, the manager can distribute work to off hours (using batch queues) and limit the number of concurrent interactive users (using DCL commands or Authorize utility commands).

The AUTOGEN command procedure, run at system installation, sets a number of system parameters by detecting devices installed in a configuration. AUTOGEN can also be run on a regular basis using a command procedure. The AUTOGEN feedback option generates a report of recommended parameter settings for system tuning. The System Management utility can also be used to set dynamic system parameter values.

OpenVMS can adjust itself dynamically during operation. Built-in adjustment capabilities optimize the OpenVMS system to meet the needs of particular environments (such as timesharing, transaction processing, batch processing, or real-time processing) and different types of application work load (such as business or engineering).

OpenVMS provides a dynamic activity-based cache for frequently accessed blocks of a file, improving I/O performance. The cache automatically sizes itself based on available memory. The virtual I/O cache works on all OpenVMS configurations, from single-node systems to mixed-architecture VMSclusters.

In addition to performance capabilities provided by OpenVMS operating system software, the system manager can optionally employ other performance management software, particularly useful in large, complex environments such as production system environments. Optional performance management software suited for use in the OpenVMS production server environment is described in Section 7.1.3.3.

3.2.2.5 Managing Batch and Print Queues

OpenVMS facilities to perform batch and print operations involve the creation of queues and the setup of spooled devices to allow processing of batch, timesharing, and real-time jobs. The OpenVMS operating system provides generic queues that hold batch or print jobs until appropriate execution queues become available. An execution queue is the queue through which the job is actually processed or executed. In a VMScluster, the queue management workload can be divided across multiple queue managers on different nodes.

The system manager can use DCL commands to regulate the number of queues and the number of batch jobs in the queue that can execute concurrently. The manager can set queue attributes, start and stop queues and the jobs in queues, and perform other queue maintenance operations, as shown by the examples in Table 3–4.

Table 3–4 Examples of OpenVMS Queue Management Commands

System Management Task	DCL Command
Delete a print or batch queue and all the jobs in the queue	DELETE/QUEUE command
Create or initialize queues and assign names and attributes to the queues	INITIALIZE/QUEUE command
Start or restart the specified queue after it has been initialized	START/QUEUE command
Pause the specified queue and suspend all the jobs currently executing on the queue	STOP/QUEUE command
Stop all queues on a node	STOP/QUEUE/ON_NODE command
Change the attributes of the queue	SET QUEUE command
Display information about queues and the jobs currently in queues	SHOW QUEUE command
Display information about a user's batch and print jobs or about specific job entries	SHOW ENTRY command
Queue one or more batch jobs to a batch queue	SUBMIT command
Queue one or more files for printing to an output queue	PRINT command

3.2.3 Management Software for Specific Environments

A system manager can configure the OpenVMS Alpha or OpenVMS VAX system as a node in a network and can establish a VMScluster or VAXcluster system configuration. In addition, a manager can set up and manage special configurations and special processing environments. The optional POSIX for OpenVMS product is installed and managed separately.

To prepare the OpenVMS system for the network environment, the system manager chooses the networking software to be used, selects a node name and network address, and makes the necessary hardware connections to the network.

The manager performs network installation and configuration procedures using the appropriate networking software tools, and starts the network software running on the system. The system manager can use the following commands to manage different network services (such as DECnet): SET NETWORK, SHOW NETWORK, START NETWORK, and STOP NETWORK.

To keep the network software running, the system manager uses network tools to perform basic problem solving for the local node. Overall network management is summarized in Section 2.4.2 and network management tasks and tools are described in Section 6.2.1. Refer to the appropriate networking product documentation for specific information on installing, configuring, and managing networking software.

To establish a VMScluster system, the system manager prepares the cluster operating environment, sets up cluster disks and tapes and cluster queues, and builds the cluster. The interactive CLUSTER_CONFIG command procedure permits the manager to directly configure the cluster. The command procedure can add or remove a computer from the cluster, change a computer's characteristics, or create a duplicate system disk.

Managing the VMScluster system is similar to managing an individual OpenVMS system. The same tools and DCL commands are used to perform management tasks, with the exception that the System Management utility (SYSMAN) is used to perform clusterwide management. SYSMAN provides the ability to centralize the management of a VMScluster system by defining and then managing the entire cluster from a single node. However, a VAX CPU and an Alpha CPU cannot boot from a common system disk. In a VMScluster that includes both Alpha and VAX nodes, VAX and Alpha CPUs must boot from their own boot servers. The system manager defines a system management environment in which operations performed from the local Alpha or VAX system can be executed on all other systems in the defined environment. SYSMAN accepts the following sets of commands as well as most DCL commands: CONFIGURATION, STARTUP, PARAMETERS, and DISKQUOTA. In addition, the dynamic Show Cluster utility displays the status of VMScluster or VAXcluster hardware components and communication links. See Section 3.3.1 for a description of managing security in a VMScluster environment.

Other possible management responsibilities for the OpenVMS system manager may include:

- Managing special system configurations (such as terminal and InfoServer connections). The LAT Control Program (LATCP) utility is used to configure and control the LAT protocol on OpenVMS host systems; LAT supports communication with terminal servers. The LAD Control Program (LADCP) configures and controls the LAD protocol for OpenVMS host systems. The LAD protocol permits a user to access InfoServer discs (CD–ROMs) and

other disks and tapes as though they were locally connected to the OpenVMS system; therefore, several OpenVMS nodes can share the same disk media.

- Managing special processing environments (such as transaction processing, symmetric multiprocessing, or vector processing environments). For example, the system manager can use the Log Manager Control Program (LMCP) utility to create and manage log files that are used by transaction managers (see Section 7.2.3) and can use the Monitor utility to monitor the status of transactions executing on the system.
- Installing and managing the POSIX for OpenVMS environment. For detailed information about installing and managing POSIX on an OpenVMS system, refer to the *POSIX for OpenVMS Installation and System Management Guide*.

3.2.4 OpenVMS Management Station

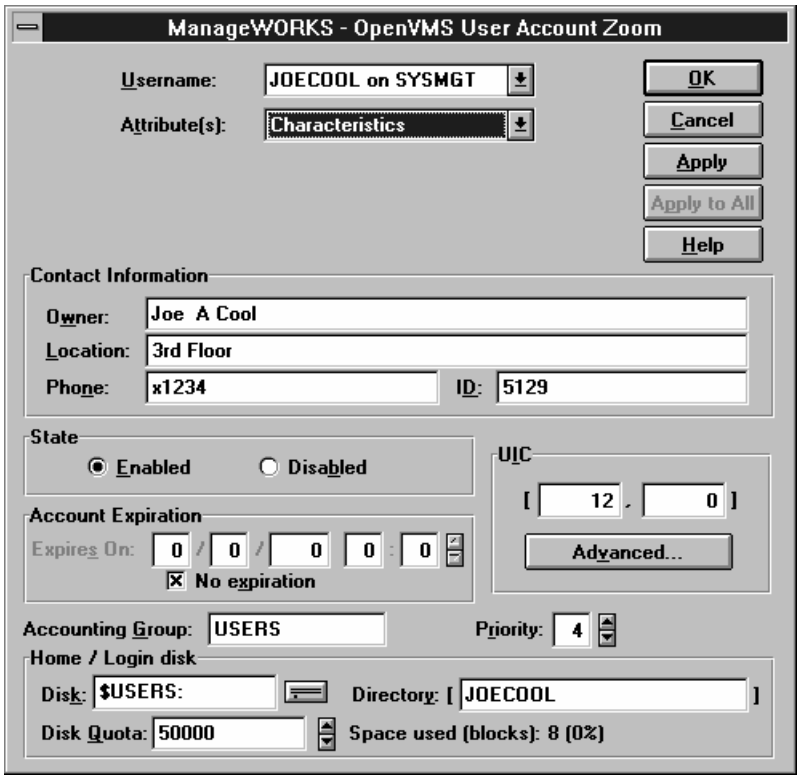
The OpenVMS Management Station software enables management of one or more OpenVMS systems from a Microsoft Windows based PC. The software presents a hierarchical view and provides the ability to manage multiple OpenVMS systems from a single point of control. Users can view, organize, and manage objects in a way that is meaningful to them.

The first version of the Management Station software provides for OpenVMS user account management. The software is designed to be able to support additional system management functions in subsequent versions.

OpenVMS Management Station provides an easy-to-use interface to the process of managing OpenVMS accounts, eliminating the need to use multiple OpenVMS utilities (for example, DCL, the Authorize utility, and DISKQUOTA). When creating an account on multiple systems, OpenVMS Management Station can add a UAF entry, grant rights identifiers, create an OpenVMS directory, set a disk quota, and carry out other steps, for each instance of an account. Figure 3–2 shows a sample OpenVMS Management Station screen. The OpenVMS Management Station performs the following account management operations:

- Creates, modifies, and deletes user accounts
- Renames user accounts
- Displays user account attributes

Figure 3–2 Sample OpenVMS Management Station Screen



The OpenVMS Management Station client is installed on a PC, and the server component is installed on each OpenVMS system to be managed. The client on the PC supports DECnet Phase IV and TCP/IP connections to the servers. The servers on OpenVMS systems also support the use of DECnet/OSI Phase V between servers that use Phase IV alias names.

Extensive Microsoft Windows help files describe the software features and functions and include step-by-step instructions and numerous examples. For summary information about using the software, see the *OpenVMS Management Station Overview and Release Notes*.

3.3 OpenVMS System Security

Safeguarding the information processed and stored on the OpenVMS system can be an important consideration, especially as many system environments become more open and distributed. OpenVMS provides an extensive range of built-in features that can be used to secure the OpenVMS computing environment, ensuring that information is available only to those who need it and that unauthorized access is prevented.

The following sections describe the security environment and security standards supported by OpenVMS Alpha and OpenVMS VAX. They identify the primary security features and security management software and tools.

3.3.1 OpenVMS Security Environment and Standards

Security concerns apply to whole environments, such as offices, because information processing no longer necessarily occurs only in limited, centralized datacenters. OpenVMS security features are designed to protect systems and information in any configuration or environment and apply to VMScluster systems as well as single standalone OpenVMS VAX or Alpha systems.

The C2 level of security is defined in the Department of Defense System Trusted Computer System Evaluation Criteria (TCSEC), published by the National Computer Security Center (NCSC). The C2 criteria are the security capabilities required by government and defense contractors and also by many commercial customers.

OpenVMS VAX Version 6.0, which was evaluated by NCSC as meeting C2 criteria, has been upgraded to OpenVMS VAX Version 6.1 using RAMP. (RAMP is the NCSC Ratings Maintenance Process.) The security requirements for a class C2 system are built into the OpenVMS operating system and include extensive auditing capabilities and discretionary access controls.

Additionally, SEVMS VAX Version 6.0, which was evaluated by NCSC as meeting B1 criteria, has been upgraded to SEVMS VAX Version 6.1 using RAMP. Class B1 functionality includes mandatory access controls, labeled object protection, and additional auditing.

OpenVMS AXP Version 6.1 and SEVMS AXP Version 6.1 are participating in the RAMP for C2 and B1 ratings, respectively.

In terms of security, the system is the computing and communication environment over which the manager has some control. The system protects everything inside the system. The subset of OpenVMS that is secure is called the Trusted Computing Base. It includes the executive and file system, other components that do not execute in user mode, system programs, and related system management utilities. The security domain controls access mediation, resource allocation, authorization, and the auditing subsystem.

OpenVMS networking connections are not part of the security domain. The DECnet software used to link the systems in a network provides separate built-in security controls. Optional encryption services protect messages transmitted between systems over the local area network (LAN) (as described in Section 6.2.3).

3.3.2 OpenVMS Security Management Software

The OpenVMS operating system provides built-in security features that can be implemented in a flexible manner. File protection can be extended to protect all, none, or some of the files. Passwords can be general, screened, locked, or eliminated. The auditing subsystem can be used to warn a system manager, operator, or security administrator of attempted system intrusions or unauthorized access to system files or other objects.

Selective use of security features permits establishment of a high degree of security for sensitive data, while minimizing or eliminating control procedures for less essential data. Table 3–5 summarizes some significant security features provided by OpenVMS VAX and OpenVMS Alpha.

Table 3–5 OpenVMS Security Features

OpenVMS Security Feature	Values or Description
Login classes	Local, remote, dialup, network, batch, process
Access modes	Interactive and noninteractive
Password authentication and management	Security controls involving password length, password generation, system passwords, password dictionaries, password histories
User accounts as defined in the UAF	Regular, captive, and restricted accounts
Intrusion detection and evasion	Mechanisms for detecting intrusion attempts and features for automatic evasion of intrusions
Categories of file protection	System, owner, group, world
Rights for each category of file protection	Access rights: read, write, execute, delete, control
Privileges	Granted to users according to their need to access system functions
Protection based on user identification code (UIC)	Determined by owner UIC and a protection code; controls access to objects (for example, files, directories, volumes, queues)
Access control list (ACL) protection	Matches specific access to specific users or groups of users for each object
Proxy access definition	Permits a user from a remote node to log in to a local node as if the user owned an account on the local node
Auditing and logging	Comprehensive auditing subsystem
Secure terminal path	Capability designed to thwart ''password grabbers''
High-water marking	Technique for discouraging disk scavenging

The OpenVMS system provides mechanisms for controlling user access, securing data and resources, and creating an audit trail.

The OpenVMS system manager can use OpenVMS security tools to define levels of protection for and control access to memory, files, devices, and other OpenVMS security objects. The manager can establish a security database that includes the following controls:

- For security subjects (user processes, jobs, and applications that need to access security objects), the manager sets up the system authorization file (SYSUAF.DAT), the user rights database file (RIGHTSLIST.DAT), and the network proxy database file (NET$PROXY.DAT)[1]
- For security objects, the manager specifies protection codes, the UIC assigned to the owner of the object, and the access control list (ACL) that defines the access granted to specific users. (Examples of security objects are files, devices, volumes, queues, global sections, logical name tables, common event flag clusters, resource domains, and capabilities.)
- The manager establishes an audit trail, a listing of enabled security-related events, that can be used to monitor system activity.

[1] For backward compatibility, the DECnet Phase IV NETPROXY.DAT file is also supported.

Within the security domain, authentication is the procedure for verifying a person's identity, through passwords. Mechanisms for identification on an OpenVMS system include login to a user account and proxy login. The OpenVMS system manager uses the Authorize utility to establish user accounts and controls passwords in the SYSUAF (see Section 3.2.2).

The Authorize utility permits the system manager to set up or modify an account that matches the individual user's needs. An interactive user who performs general work on the system needs an individual account and a file directory. Other users may require only a limited-access account to a restricted system environment, for example, to perform routine tasks (such as shop-floor operations), run batch operations during unsupervised periods, or run an application program containing private information. Turnkey or captive accounts limit the activities of the user and deny the user access to the DCL command level. Restricted accounts have features similar to those of a captive account but permit access to the DCL command level after login command procedures are executed (for example, to permit a user to access electronic mail).

A system manager can use either a DCL command-line interface or a Windows based OpenVMS Management Station (described in Section 3.2.4) to manage user accounts on multiple OpenVMS systems (as in a VMScluster).

OpenVMS supports discretionary access controls that permit individually named users to be either included or excluded for accessing a certain file or achieving particular forms of access.

In addition, the system manager can set up a certain application as a protected subsystem. In the protected subsystem, data files and resources are controlled by the subsystem and data can be accessed only by running the subsystem. When it is run, the subsystem causes the process running the application to be granted additional identifiers. The system manager grants the identifiers to the people who will serve as managers of the subsystems.

The Security Server process, created during system startup, creates and manages the system's intrusion database and maintains the network proxy database file. The intrusion database is used by the system to keep track of failed login attempts. The information is scanned during process login to determine if the system should use restrictive measures to prevent access to the system by a suspected intruder. The network proxy database file is used during network connection processing to determine if a specific remote user may access a local account without using a password. The network proxy database file is managed by the Authorize utility.

Security auditing facilitates tracking of sensitive system objects. Auditing enables the manager to monitor system activities and prevent unauthorized access. The OpenVMS auditing mechanism allows users and security managers to record security-related events on an audit log file and possibly send them to an operator terminal designated as a security alarm console. The system manager can set security alarms for events like login failures, authorization changes, and file access. The manager or security administrator can use the SET AUDIT command to select events to be audited. The administrator uses the Audit Analysis utility to extract audit trail information from the audit log file.

3.4 Optional OpenVMS System Integrated Software

The following optional software capabilities are integrated into the OpenVMS Alpha and OpenVMS VAX operating systems:

- VMScluster system software
- Volume-shadowing software
- RMS Journaling software

The following sections summarize the functions supplied by these OpenVMS integrated software products.

3.4.1 VMScluster Software

The VMScluster system provides a highly integrated OpenVMS computing environment distributed over multiple CPUs, including Alpha CPUs and, optionally, VAX CPUs. The VAXcluster system is an integrated environment distributed over multiple VAX CPUs. Similar software is used in VMScluster and VAXcluster configurations.[2] The individual CPUs that are members of a VMScluster system can be connected by supported interconnects (as described in Section 1.3.3).

OpenVMS system managers can tailor the VMScluster operating environment to create a common environment (with the same resources available on all members) or a multiple environment (with different resources shared by specific groups of members of the VMScluster or, possibly, one member providing special-purpose functions).

In any VMScluster system, users can share computing, disk storage, and batch and print job processing resources. The ability to share resources facilitates work-load balancing because work can be distributed across the cluster. Resources can be added or removed without disrupting normal cluster operation.

Members of a VMScluster system can share processing resources, data storage, and queues under a single OpenVMS security and management domain. Applications run on one or more CPUs in the cluster, accessing shared resources in a coordinated manner. Most cluster resources can be shared, but user processes and memory are CPU specific, and each member can boot or fail independently. Each CPU in a VMScluster boots from its own boot server. In a mixed-architecture VMScluster, the boot server for a CPU must be of the same architecture as the CPU.

The software components used to implement VMScluster communication and resource-sharing functions always run on each member of the cluster. If one member fails, the VMScluster system continues operating because the components still run on the remaining members. These software components are summarized in Table 3–6.

[2] For convenience, this section refers to VMScluster configurations. Any exceptions that apply to a specific cluster configuration are noted.

Table 3–6 VMScluster Software Components

VMScluster Component	Function
System Communications Services (SCS) software	Implements intercomputer communication, according to the Digital System Communications Architecture
Connection manager	Dynamically defines the VMScluster system and coordinates participation of members in the cluster; uses SCS to provide an acknowledged message delivery service for higher OpenVMS software layers; maintains cluster integrity when computers join or leave the cluster
VMScluster distributed file system	Allows all computers to share mass storage, whether the storage device is connnected to an HSC subsystem or to a computer; used to provide the same access to disks and files across the cluster that is provided on a standalone OpenVMS computer
Queue manager	Makes batch and print queues available across the cluster
Distributed lock manager	Used for synchronization functions by cluster facilities and cluster applications developers; implements system services to provide clusterwide synchronization of access to resources by allowing the locking and unlocking of resource names; provides a queueing mechanism so that processes can be put into a wait state until a particular resource is available; supports clusterwide deadlock detection
MSCP and TMSCP servers	Implement the disk mass storage control protocol and tape mass storage control protocol for communicating with disk and tape controllers, respectively; make locally connected disks and tapes available across the cluster

VMScluster systems can boot satellites using either DECnet or the LANCP (LAN control program) facility. With OpenVMS Version 6.2 and later versions, DECnet software (either DECnet Phase IV or DECnet/OSI) is required only when the applications perform node-to-node communication using DECnet mailboxes or when the MONITOR/CLUSTER utility is used.

DECnet supports the VMScluster alias node identifier (a node name or address): a special node identifier common to some or all nodes in the cluster. The alias node identifier permits users outside the cluster to address the cluster as though it were a single node.

The LANCP utility can be used to boot VMScluster satellite nodes in configurations that are not using DECnet. LANCP and LANACP (LAN Ancilliary Control Process) images provide a general-purpose MOP booting service for satellite nodes.

System managers control how jobs share batch processing and printer resources by setting up and maintaining clusterwide generic queues. A generic queue holds a job that will execute on an execution queue on a specific node when the node is available to process jobs. The strategy for setting up and managing the generic queues determines how well work loads are matched to available resources. The clusterwide queue manager process accesses the clusterwide queue database for all processes in a cluster. Job controllers, user processes, and print symbionts all communicate directly with the centralized queue manager through a shared interprocess communication link.

VMScluster systems can make disk and tape storage resources accessible to all VMScluster members. A cluster-accessible storage device can be used directly by multiple members of the cluster. OpenVMS MSCP and TMSCP server software can make disks and tapes accessible to members. The TMSCP server can make locally connected SCSI tapes available across a VMScluster.

Cluster-accessible disks offer the following advantages:

- More efficient use of mass storage, because more than one member can use the same disk.
- Access by users to their default work disks when logging in to any member on which the disks are accessible.
- Clusterwide file sharing. Because members can share common versions of files, updates to a file are made only once to a single copy of the file.
- Implementation of clusterwide queues. Batch and print jobs can be processed on any member that has access to the necessary disks.

Some VMScluster systems include HSC hierarchical storage controller subsystems, which are self-contained, intelligent mass storage subsystems that enable VMScluster members to share DSA disks and DSA tapes. HSC disk configurations provide flexibility, expansion potential, and maintenance and backup capability.

The OpenVMS distributed lock manager and the distributed file system enable the development of distributed applications that work from the same data, which is itself distributed across the VMScluster.

Disk data can be replicated between VMScluster systems for better read performance and higher availability using volume shadowing, as described in the following section.

3.4.2 Volume-Shadowing Software

Volume-shadowing software for OpenVMS enhances data availability by duplicating all data written to disk onto compatible disk volumes, called shadow sets. A shadow set consists of one, two, or three disk volumes of the same model, referred to as shadow-set members. The volume-shadowing software can read data from any member of the shadow set.

Because volume shadowing simultaneously records data on more than one disk, the duplication of data ensures that data is consistently available. If one disk becomes unreadable because of normal media deterioration, communication path failure, or controller or device failures, processing continues with another disk in the shadow set. The process of shadowing is invisible to end users and applications. Disks can be added or removed without affecting the user or the application.

Volume shadowing is controller independent and supports shadowing of VMScluster devices. (Phase I volume shadowing, supported only on OpenVMS VAX systems, performs shadowing of disk devices connected to an HSC.) Phase II volume shadowing, which runs on OpenVMS Alpha and OpenVMS VAX systems, supports many more disk controllers and devices in a wider range of configurations. Host-based volume shadowing also supports the use of StorageWorks RAID Array Subsystems. Using volume shadowing in a VMScluster system with multiple controllers ensures a high degree of data availability (see Section 7.1.2.1).

Shadow sets are created with the Mount utility or the $MOUNT system service (which are used to make a disk volume available for processing). A disk is added to the shadow-set by means of the MOUNT command or the $MOUNT system service. Volume-shadowing routines ensure that, within a reasonable time, the newly added shadow set member is made identical to the other member or members of the set. A disk is removed from the shadow set by operator command or automatically, if the disk becomes inoperative.

3.4.3 RMS Journaling Software

RMS Journaling, which runs on OpenVMS VAX and OpenVMS Alpha systems, is an optional software tool that maintains the data integrity of RMS files if any of a number of failures occur. It helps to protect RMS data from becoming lost or inconsistent. RMS Journaling supports distributed transactions through the use of DECdtm (see Section 3.1.6) and provides support for programs that use multiple concurrent transactions. A transaction is a series of RMS record operations made on one or more files. RMS journaling helps prevent data from becoming inconsistent due to the incomplete execution of a transaction.

RMS Journaling provides three forms of journaling:

- After-image journaling provides the means to redo a series of modifications to a data file, enabling the recovery of lost or corrupted files. After-image recovery restores the contents of the file from the point of its latest backup copy.
- Before-image journaling provides the means to undo a series of modifications to a data file, in the event that a file is updated with erroneous data. It also permits automatic rollback to the last consistent state in the event of an error or a failure.
- Recovery-unit journaling maintains transaction integrity by preventing partial completion of transactions.

4

Development on OpenVMS Systems

An essential feature of the OpenVMS operating system is its support of a rich environment for developing software application programs. The programming software integrated in the OpenVMS system provides the tools required to effectively develop new software applications. The developer also has the option of using additional powerful tools to enhance the productivity of software development in the OpenVMS environment.

This chapter summarizes the primary program development features available on all OpenVMS VAX and OpenVMS Alpha systems and indicates any programming features specific to a particular operating system.

The chapter introduces the common programming environment and presents brief functional descriptions of the OpenVMS programming tools, as well as POSIX for OpenVMS programming capabilities. Using optional software development tools running on OpenVMS systems in the context of distributed multivendor environments is covered in Section 6.3.5.

4.1 Common Programming Environment

The OpenVMS system supports a flexible programming environment that offers a wide range of tools and resources to support efficient program development. The common programming environment permits the development of mixed-language application programs and portable programs, as well as application programs with distributed functions that run in client/server environments.

In the common programming environment, programmers can use OpenVMS resources to perform such tasks as:

- Creating, controlling, and deleting processes
- Communicating with other components
- Sharing resources
- Implementing input/output procedures
- Using security features
- Managing memory
- Managing files
- Synchronizing events
- Providing for condition handling
- Calling utility routines

The components of an OpenVMS application are the main program, shared libraries, functional routines, and a user interface. Software tools that support development of applications in the OpenVMS programming environment include:

- Language compilers, interpreters, and assemblers
- Linkers and debuggers
- Text processors and other program development utilities
- Callable system routines such as run-time routines, system services, and other utility routines
- Record Management Services (RMS) routines and utilities

Optional software development tools that run on the OpenVMS system enhance programmer productivity, saving programming time and promoting the development of error-free code. OpenVMS supports optional integrated software products that enhance program development capabilities in an organization. These software development products can make use of middleware services, which facilitate the development of applications for multivendor networks. Optional software development products are discussed in Section 4.7, and use of these optional tools in a distributed multivendor environment is covered in Section 6.3.5.

4.1.1 Programming to Standards

Coding of programs for the OpenVMS environment and for other environments involves conforming to software development standards. OpenVMS standards that define modular programming techniques and procedure calling and condition handling practices pertain to applications specific to OpenVMS. IEEE and international standards apply to applications developed on OpenVMS that are designed to run on other systems as well as OpenVMS. Refer to Appendix A for a list of standards supported by OpenVMS.

4.1.1.1 Common Environment for Writing Code

OpenVMS software programmers can write code in a common environment, following standard OpenVMS modular programming practices. This standard approach establishes the minimum criteria necessary to ensure the correct interface at the procedure level between software written by different programmers. If all programmers coding OpenVMS applications follow this standard approach, modular procedures added to a procedure library will not conflict with other procedures in the library. Standard modular programming practices apply to OpenVMS programs that have a public entry point. For details of this standard approach, see the *Guide to Creating OpenVMS Modular Procedures*.

4.1.1.2 Common Language Environment

The OpenVMS system supports a common language environment, which permits using a mixture of languages in programming. A program written in any of the programming languages supported by OpenVMS can contain calls to procedures written in other supported languages. Mixed-language programming is possible because all supported languages adhere to the OpenVMS calling standard. This standard describes the techniques used by all supported languages for invoking routines and passing data between them. It also defines the mechanisms that ensure consistency in error and exception handling routines, regardless of the mix of programming languages in use. Information about the calling standard appears in the *OpenVMS Calling Standard*, and a description of how to use the

calling interface is given in *OpenVMS Programming Interfaces: Calling a System Routine*.

4.1.2 Developing Portable Programs

A software program that is portable can be ported from one computer system to another. OpenVMS programmers can design and develop programs to run on different platforms and execution environments. The primary concerns in designing portable applications include:

- Using modularization and abstraction to organize software
- Avoiding platform-specific features

POSIX for OpenVMS software that conforms to IEEE POSIX standards (see Section 2.1.3.1) allows OpenVMS users to develop applications that can be ported to other systems that support POSIX. The source code of an application that conforms to POSIX standards can be ported to another POSIX conforming system and compiled and linked on that system to produce an executable image. The programming interface for POSIX for OpenVMS is described in Section 4.4 and the user interface in Section 5.3.

OpenVMS users can convert application programs that run on OpenVMS VAX systems to run on OpenVMS Alpha systems by recompiling and relinking or by translating. A single application can include both native images (those that were recompiled and relinked) and translated images.

- The most effective way to convert a program from OpenVMS VAX to OpenVMS Alpha is to recompile the source code using a native OpenVMS Alpha compiler and then to relink the object files and shareable images using the OpenVMS Linker. This method produces a native OpenVMS Alpha image that can take full advantage of the speed of the Alpha system.
- The alternative method involves translating user-mode OpenVMS VAX images to run on OpenVMS Alpha systems. The translated image provides a high degree of OpenVMS VAX compatibility but does not provide the same high performance as a recompiled image. Translation is used primarily when recompilation is not practical or possible.

Software tools for converting programs from VAX platforms to Alpha platforms are described in Section 4.2.2.

For a detailed discussion of ensuring the portability of an application developed on an OpenVMS VAX system to run on an OpenVMS Alpha system, see *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*. This manual highlights OpenVMS Alpha features that contribute to portability, indicates differences in the programming environments, and provides guidelines for developing new programs intended to run on both OpenVMS VAX and OpenVMS Alpha systems.

4.2 OpenVMS Programming Software

This section describes the integrated programming tools available on the OpenVMS operating system to help implement software development.

The phases of a typical software development life cycle can include proposal of the concept; formulation of requirements and specifications for the software product; design, implementation, and testing of the software; and integration and maintenance of the product. Implementing the software product involves building and modifying source code modules and compiling, linking, and executing the resulting images. Testing involves refining code to optimize performance.

As part of the software development life cycle, OpenVMS operating system components and optional software products that run on OpenVMS are used to develop applications. Some of the major OpenVMS programming software components, such as editors and utilities, are listed in Table 4–1. Programming language software supported by OpenVMS is described in Section 4.2.2. Optional program development software tools that run on OpenVMS are described in Section 4.7.

Table 4–1 OpenVMS Programming Software

Type of Software	OpenVMS Software Components
Text processors	DEC Text Processing Utility/Extensible Versatile Editor (DECTPU/EVE) EDT Editor Text Editor and Corrector (TECO) vi, ed, and ex editors (POSIX) DEC Language-Sensitive Editor/Source Code Analyzer (LSE/SCA)
Major programming utilities	Linker OpenVMS Debugger Delta/XDelta Debugger OpenVMS Alpha System-Code Debugger‡
Other program development utilities	Command Definition utility Librarian utility Message utility Patch utility† SUMSLP utility National Character Set utility System Dump Analyzer[1] POSIX for OpenVMS utilities
Callable system routines	Run-time library routines System services Utility routines Record Management Services (RMS) routines and utilities

[1]Different versions run on the VAX and Alpha platforms.

†VAX specific.

‡Alpha specific.

The commands used to invoke some of the programming utilities (for example, linker, debugger, LIBRARIAN) vary slightly for VAX and Alpha platforms.

4.2.1 Creating Program Source Files

OpenVMS text-processing utilities can be used to create and modify program source files. The DEC Text Processing Utility (DECTPU) is a high-performance text processor that can be used to create text-editing interfaces such as EVE. DECTPU includes a high-level procedure language with its own compiler and interpreter, as well as the customizable EVE editing interface. DECTPU features multiple buffers, windows, and subprocesses, and provides for text processing in batch mode. The EDT editor is an interactive text editor that provides editing in keypad and line modes. EDT supports multiple buffers, startup command files, and journaling. In general, the EVE editing interface offers more capability than EDT for complex editing tasks. (Use of the EDT and EVE editors for editing text files is described in Section 5.5.4.)

TECO is a traditional character-oriented text-editing program that runs under both OpenVMS VAX and OpenVMS Alpha. It can be used to edit program sources and other ASCII text.

The vi editor is a display-oriented interactive text editor used in the POSIX for OpenVMS environment. POSIX also supports the ed and ex editors.

Other optional tools for creating source files on OpenVMS systems are available separately or as part of the Digital software development environment (see Section 6.3.5). The DEC Language-Sensitive Editor/Source Code Analyzer (LSE/SCA) provides a multilanguage, multivendor editor for program development and maintenance and also supplies cross-referencing features and the capability to analyze source code.

4.2.2 Creating Object Files

OpenVMS supports a variety of optional language compilers, interpreters, and assemblers that translate source code to object code (in the form of object modules). These language implementations adhere to industry standards, including ISO, ANSI, X/Open, and POSIX standards as well as U.S. Federal Information Processing Standards (FIPS) and Military Standards (MIL-STD), as applicable.

Table 4–2 lists language compilers, interpreters, and assemblers supported in the OpenVMS VAX and OpenVMS Alpha environments. Names that begin with DEC refer to software available on Alpha and VAX platforms unless otherwise indicated.

Table 4–2 Compilers, Interpreters, and Assemblers

Language	Characteristics
DEC Ada	Complete production-quality implementation of Ada language; fully conforms to ANSI and MIL-STD standards; has Ada validation
VAX APL	Interpreter with built-in editor, debugger, file system, communication facility
VAX BASIC	Used as either an interpreter or a compiler; fully supported by the OpenVMS debugger; fully reentrant code
DEC BASIC for OpenVMS Alpha	An optimizing compiler; highly compatible with VAX BASIC; no environment/interpreter support
DEC BLISS-32	Provides advanced set of language features supporting development of modular software according to structured programming concepts
DEC BLISS-64	Supports development of modular software for 64-bit programs
VAX C	Full implementation of C programming language with added features for performance enhancement in the OpenVMS environment
DEC C for OpenVMS Alpha	Compliant with ISO/ANSI-standard Programming Language C; supports 64-bit virtual addressing

(continued on next page)

Table 4–2 (Cont.) Compilers, Interpreters, and Assemblers

Language	Characteristics
DEC C++	C++ compiler with class libraries, a new C Run-Time Library, and debug support; facilitates object-oriented program design
DEC COBOL	Compliant with ANSI-standard COBOL; includes as enhancements screen-handling, file-sharing, and report-writing facilities
VAX DIBOL	Designed for interactive data processing; includes a compiler, debugger, and utility programs for data handling, data storing, and interprogram communication
DEC Fortran for OpenVMS VAX	Supports ANSI-standard FORTRAN–77 with many industry-leading extensions; conforms to FIPS standards; has a high optimization compiler and takes full advantage of features of OpenVMS environment
DEC Fortran for OpenVMS Alpha	Supports ANSI-standard FORTRAN–77, nearly all DEC Fortran for OpenVMS VAX extensions, and other language features, including recursion
DEC Fortran 90	Supports ANSI-standard FORTRAN-90 for high-performance capabilities using parallel and clustered resources
VAX MACRO	Assembly language for programming the VAX computer under the OpenVMS operating system; uses all OpenVMS resources; supports large instruction set enabling complex programming statements
MACRO-32 Compiler	Available on OpenVMS Alpha systems to port existing VAX MACRO code to an Alpha system
MACRO-64 Assembler	Available on OpenVMS Alpha systems; a RISC assembly language that provides precise control of instructions and data
DEC OPS5	Highly efficient language for implementing expert systems; used to apply artificial intelligence technology to production systems
DEC Pascal	Supports ANSI-standard Pascal features and added features using character instruction sets and OpenVMS virtual memory
VAX PL/I	Includes a compile-time preprocessor that allows language extension and conditional compilation
DEC PL/I for OpenVMS Alpha	Used for complex programs; supports many data types including floating-point

OpenVMS also serves as the host operating system for Ada cross-compilers, cross-assemblers, and cross-development tools that conform to MIL-STD standards.

DECmigrate for OpenVMS AXP is a Digital layered product that translates OpenVMS VAX images into OpenVMS Alpha images. It also can be used to analyze code to determine how easy or difficult it would be to migrate it. The VAX Environment Software Translator (VEST) component of DECmigrate produces the translated images that can be executed on OpenVMS Alpha. The *DECmigrate for OpenVMS AXP Systems Translating Images* manual describes how to use VEST.

The following migration manuals are available with the Alpha Migration Kit:

- *Migrating an Environment from OpenVMS VAX to OpenVMS Alpha*
- *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*
- *Migrating to an OpenVMS AXP System: Porting VAX MACRO Code*

4.2.3 Creating Runnable Programs

After a program source file is coded, it must be compiled or assembled into object modules by a language processor and then linked. The OpenVMS Linker binds the object modules into an image that can be executed on the OpenVMS operating system.

The linker processes object modules and shareable image files, as well as symbol table files, library files, and options files (used to manage the linking operation and simplify the use of complex, repetitious linker operations). The most common output of the linker is an executable image of the program. The linker can also produce a shareable image, a system image, an image map, or a symbol table file to be used by other programs being linked. Certain linking tasks, such as creating shareable images, are performed differently on OpenVMS VAX and OpenVMS Alpha systems.

The Librarian utility provides for efficient storage in central, easily accessible files of object modules, image files, macros, help text, or other record-oriented information.

4.2.4 Testing and Debugging Programs

The debugger allows users to trace program execution and to display and modify register contents using the same symbols as are in the source code.

The following debugger utilities available on both the OpenVMS VAX and OpenVMS Alpha operating systems contain some system-specific features related to the platform architecture:

- The OpenVMS Debugger (debugger), which debugs user-mode code
- The Delta/XDelta Debugger (DELTA/XDELTA), which debugs code in other modes as well as user mode

The OpenVMS symbolic debugger is more useful than DELTA/XDELTA for most programs: the symbolic debugger employs user-defined symbols referring to program locations, accepts commands entered using different interfaces (keypad, command line, or file of commands), displays source code lines on the screen, has more descriptive error messages, and provides help information.

The debugger command language specified in the *OpenVMS Debugger Manual* provides more than 100 commands to control a debugging session, including these tasks:

- Control program execution on a line-by-line basis or at a user-specified breakpoint
- Display breakpoints, tracepoints, watchpoints, active routine calls, stack contents, variables, symbols, source code, and source directory search list
- Define symbols
- Create key definitions
- Change values in variables

- Evaluate a language or address expression
- Create or excute debugger command procedures

The OpenVMS symbolic debugger provides enhanced support for programs that have multiple threads of execution within an OpenVMS process, including any program that uses DECthreads for developing real-time applications.

The debugger has been modified to support debugging of programs that contain 64-bit data addresses.

An additional debugger utility is available only on an OpenVMS Alpha system: the OpenVMS Alpha System-Code Debugger, which can be used to debug non-pageable system code and device drivers. The system-code debugger is a symbolic debugger that lets the user employ the familiar OpenVMS Debugger interface to observe and manipulate system code interactively as it executes. The system-code debugger can display the source code where the software is executing and allows the user to advance by source line.

Users can perform the following tasks using the system-code debugger:

- Control the system software's execution, stopping at points of interest, resuming execution, intercepting fatal exceptions, and so on
- Trace the execution path of the system software
- Monitor exception conditions
- Examine and modify the value of variables
- In some cases, test the effect of modifications without having to edit the source code, recompile, and relink

The OpenVMS Alpha System-Code Debugger can be used to debug code written in the following languages: C, BLISS, and MACRO languages. Information about using the system-code debugger and how it differs from the OpenVMS Debugger is given in *OpenVMS Alpha Device Support: Developer's Guide*.

4.2.5 Using Other Program Development Utilities

Other OpenVMS utility programs used for program development are listed in Table 4–3. RMS utilities, which permit file analysis and tuning, are covered in Section 4.8.2.

Table 4–3 Other OpenVMS Program Development Utilities

Utility	Function
Command Definition utility	Enables an application developer to create commands with a syntax similar to DIGITAL Command Language (DCL) commands
Message utility	Permits user to create application messages to supplement the OpenVMS system messages
Patch utility†	Permits users to make temporary changes (in the form of patches) to an image file; the new version can then be run without recompiling and relinking

†VAX specific.

(continued on next page)

Table 4–3 (Cont.) Other OpenVMS Program Development Utilities

Utility	Function
SUMSLP utility	A batch-oriented editor used to make several updates to a single source file; one update program can be applied to all versions of a file
National character set utility	Permits users to define non-ASCII string collating sequences and to define conversion functions; allows an RMS indexed file to be collated using user-specified collating sequences
System Dump Analyzer utility‡	Used to determine the cause of system failures; reads the crash dump file and formats and displays it; also used to diagnose root causes that lead to an error

‡Different versions run on the VAX and Alpha platforms.

4.2.6 Managing Software Development Tasks

Optional products that run on OpenVMS systems can be used to manage the complexity of software development tasks:

- DEC Code Management System (CMS) provides an efficient method of storing project files (such as documents, object files, and other records) and tracking all changes to these files.
- DEC Module Management System (MMS) automates building of software applications.

These products are also available as part of the software development environment (see Section 6.3.5).

The optional POSIX for OpenVMS product supports the make utility, which is used to build an application in the POSIX for OpenVMS environment. This utility is analogous to MMS in the OpenVMS environment.

4.3 Using Callable System Routines

OpenVMS provides extensive libraries of prewritten and debugged routines that can be accessed by programs. Libraries specific to OpenVMS VAX and OpenVMS Alpha systems supply commonly needed routines optimized for the OpenVMS environment; these libraries include run-time library routines, system services, utility routines, and RMS services. These libraries are described in this section.

OpenVMS also supports programming for an open environment with libraries of industry-standard routines. Examples are libraries of optional routines for software products like DECwindows Motif, CDA, and Digital PHIGS and Digital GKS graphics products.

4.3.1 Using DECthreads Run-Time Library Routines

The DECthreads routine library provides a set of portable services that support concurrent programming by creating and controlling multiple threads of execution within the address space provided by a single process on an OpenVMS system. DECthreads services are based on the IEEE POSIX standard 1003.1c and are compliant with OSF Distributed Computing Environment (DCE) standards (see Section 2.2.4). On OpenVMS Alpha systems, DECthreads are layered over kernel threads (which are described in Section 3.1.1.2).

DECthreads run-time library services provide an application programming interface usable from all OpenVMS languages and an open C-only POSIX interface that adheres to the POSIX threads standard.

On OpenVMS Alpha systems, DECthreads provides support to accept 64-bit parameters.

The highly portable DECthreads interface contains routines grouped in the following functional categories:

- General threads
- Object attributes
- Mutex
- Condition variable
- Thread context
- Thread cancellation
- Thread priority and scheduling
- Debugging

For more information about threads, see the *Guide to DECthreads*.

4.3.2 Using OpenVMS Run-Time Library Routines

The OpenVMS Run-Time Library (RTL) is a set of language-independent procedures for programs to be run specifically in the OpenVMS environment. RTL routines establish a common run-time environment for application programs written in any language supported in the OpenVMS common language environment. RTL procedures adhere to the OpenVMS calling standard and can be called from any program or program module in a language supported by OpenVMS (see Section 4.2.2).

The run-time library provides general-purpose functions for application programs. Table 4–4 summarizes the groups of RTL routines.

Table 4–4 Groups of OpenVMS Run-Time Library Routines

Routine	Description
DTK$ routines	Routines that control the Digital DECtalk system
LIB$ routines	Library routines that perform generally needed system functions such as resource allocation and common I/O procedures; provide support for 64-bit virtual addressing on OpenVMS Alpha systems
MTH$ routines†	Math routines that perform arithmetic, algebraic, and trigonometric functions
DPML$ routines‡	Digital Portable Mathematics Library for OpenVMS Alpha; a set of highly accurate mathematical functions
OTS$ routines	Language-independent routines that perform tasks such as data conversion

†VAX specific
‡Alpha specific

(continued on next page)

Table 4–4 (Cont.) Groups of OpenVMS Run-Time Library Routines

Routine	Description
PPL$ routines	Parallel processing routines
SMG$ routines	Screen management routines used in design of complex images on a video screen
STR$ routines	String manipulation routines

In addition, language-specific RTL routines support procedures in Ada, BASIC, C, COBOL, Fortran, Pascal, and PL/I as well as in POSIX C. DEC C RTL routines support 64-bit programming on OpenVMS Alpha systems.

The Digital Extended Math Library (DXML) for OpenVMS is a set of libraries of mathematical subroutines that are optimized for Digital platforms. DXML includes four libraries covering the areas of basic linear algebra, linear system and eigenproblem solvers, sparse linear system solvers, and signal processing. DXML offers development and run-time options and is suitable for high-performance applications such as seismic analysis, signal and image processing, and antenna design.

4.3.3 Using OpenVMS System Services

OpenVMS system services are procedures that control resources available to processes, provide for communication among processes, and perform basic operating system functions such as I/O coordination. Application programs can call OpenVMS system services to perform the same operations that the system services provide for the OpenVMS operating system (for example, creating a process or subprocess).

At run time, an application program calls a system service and passes control of the process to it. After execution of the system service, the service returns control to the program and also returns a condition value. The program analyzes the condition value, determines the success or failure of the system service call, and alters program execution flow as required.

OpenVMS system services are divided into functional groups, as shown in Table 4–5. System services can be used to protect and fine-tune the security of the OpenVMS environment, handle event flags and system interrupts, designate condition handlers, and provide logical name services and timer services to the application. Other system services control and provide information about processes, manage virtual memory use, and synchronize access to shared resources.

Table 4–5 Groups of OpenVMS System Services

Service Group	Function
Security	Mechanisms to enhance and control system security
Event flag	Clear, set, and read event flags; place process in wait state until flags are set
AST	Control handling of software interrupts called asynchronous system traps (ASTs)

(continued on next page)

Table 4–5 (Cont.) Groups of OpenVMS System Services

Service Group	Function
Logical names	Provide a generalized logical name service
Input/output	Perform input and output operations directly at the device driver level, bypassing RMS
Process control	Create, delete, and control the execution of processes (on a clusterwide basis); permit a process on one node to request creation of a detached process on another node
Process information	Provides information about processes
Timer and time conversion	Permits scheduling of program events at specific times or time intervals; supplies binary time values
Condition handling	Designates condition-handling procedures that gain control when an exception/condition occurs
Memory management	Permits control of an application program's virtual address space
Change mode	Changes the access mode of a process
Lock management	Permits cooperating processes to synchronize their access to shared resources
DECdtm services	Provides for complete and consistent execution of distributed transactions and for data integrity
Cluster event notification†	Requests notification when a VMScluster configuration event occurs

†Alpha specific

OpenVMS I/O system services perform logical, physical, and virtual I/O and network operations, and queue messages to system processes. The $QIO system service provides a direct interface to the operating system's I/O routines. These services are available from within most programming languages supported by OpenVMS and can be used to perform low-level I/O operations efficiently with a minimal amount of system overhead for time-critical applications. (OpenVMS I/O subsystem software is described in Section 3.1.1.3.)

On OpenVMS Alpha systems, new system services provide access to 64-bit virtual address space for process private use. Additionally, new system services are available to provide high CPU performance and improved symmetric multiprocessing (SMP) scaling of I/O operations. These services exhibit high-performance gains over the $QIO service.

DECdtm services ensure consistent execution of applications on the OpenVMS operating system. In transaction processing applications, many users may be simultaneously making inquiries and updating a database. The distributed transaction processing environment typically involves communication between networked systems at different locations. DECdtm services coordinate distributed transactions by using the two-phase commit protocol and implementing special logging and communication techniques. DECdtm services ensure that all parts of a transaction are completed or the transaction is aborted. (The two-phase commit protocol supported by DECdtm is summarized in Section 3.1.6 and described in Section 7.2.3.)

4.3.4 Using OpenVMS Utility Routines

OpenVMS programs can access some OpenVMS utilities through callable interfaces. Utility routines enable programs to invoke the utility, execute utility-specific functions, and exit the utility, returning to the program. Table 4–6 lists the OpenVMS utility routines.

Table 4–6 OpenVMS Utility Routines

Routine	Utility/Facility
ACL$	Access control list editor (ACL editor)
CLI$	Command Definition utility (CDU)
CONV$	Convert and Convert/Reclaim utilities (CONVERT and CONVERT/RECLAIM)
DCX$	Data Compression/Expansion facility (DCX)
EDT$	EDT editor
FDL$	File Definition Language utility (FDL)
LBR$	Librarian utility (LIBRARIAN)
LGI$	LOGINOUT routines
MAIL$	Mail utility (MAIL)
NCS$	National Character Set utility (NCS)
PSM$	Print Symbiont Modification facility (PSM)
SMB$	Symbiont/Job-Controller Interface facility (SMB)
SOR$	Sort/Merge utility (SORT/MERGE)
TPU$	DEC Text Processing Utility (DECTPU)

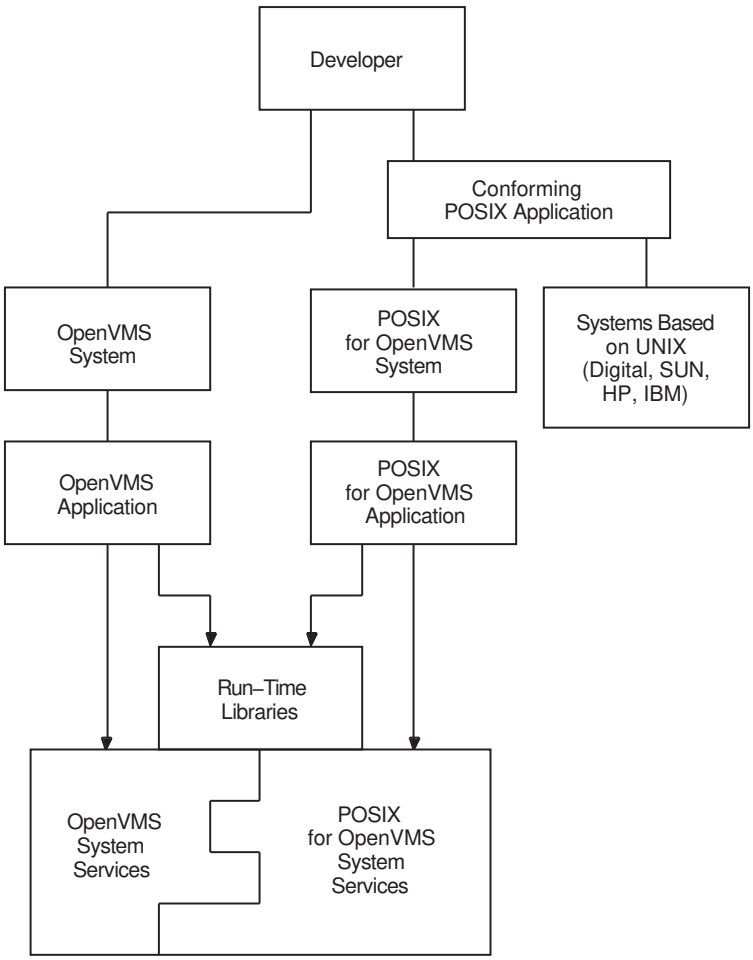
An optional, portable library of user-callable routines can be used to perform high-performance sorting on OpenVMS Alpha systems. The high-performance sort supports a subset[1] of the functionality present on the OpenVMS Sort/Merge utility, using the callable interface to the SOR$ routine. The high-performance sort/merge provides better performance for most sort and merge operations.

4.4 POSIX Programming on an OpenVMS System

The POSIX for OpenVMS product offers customers the capability to develop and run open, portable applications on OpenVMS VAX and OpenVMS Alpha systems. Applications written to POSIX and X/Open standards and draft standards (see Section 2.1.3.1) are portable across a wide range of systems that support those same standards, including systems that do or do not support UNIX. Application developers can develop and deploy their applications on any system that conforms to POSIX standards, including OpenVMS VAX and OpenVMS Alpha (see Figure 4–1). POSIX application portability is discussed in Section 2.1.3.1.

[1] Available with OpenVMS Alpha Version 7.0; full functionality is planned for a future version of OpenVMS Alpha.

Figure 4–1 Developing POSIX Applications



4.4.1 POSIX Applications Using OpenVMS Capabilities

Applications conforming to POSIX standards can coexist and interoperate with traditional OpenVMS applications on the same system. POSIX applications, when run in the OpenVMS environment, are able to take advantage of OpenVMS capabilities such as VMSclusters and volume shadowing, as well as file and data protection, security, and SMP. POSIX applications can also optionally call OpenVMS services and libraries, such as file journaling. In addition, POSIX applications can use other standards-based tools such as Motif and TCP/IP Services.

Users can log in to POSIX for OpenVMS either directly or indirectly during an OpenVMS session. OpenVMS and POSIX for OpenVMS services are mutually accessible (see Section 5.3). POSIX for OpenVMS program developers who work at the DCL level can use the OpenVMS program development environment tools, including the C compiler, the OpenVMS Linker, the OpenVMS Debugger, and their choice of editor, as well as optional application-building tools that run on OpenVMS. Developers who work at the POSIX for OpenVMS shell level can use the environment defined in POSIX standard 1003.2 for linking and compiling programs, building applications, and archiving library entries. They can also

use the vi utility and other software development utilities defined in the POSIX 1003.2 standard.

4.4.2 POSIX for OpenVMS Programming Interface

POSIX for OpenVMS application programs are written using the C language and functions defined by the POSIX, ISO C, and X/Open standards and draft standards.

POSIX for OpenVMS implements the POSIX 1003.1 standard, which incorporates the functionality described in the ISO C standard as well as additional system services not covered by ISO C. Specifically, POSIX 1003.1 covers the following:

- Process primitives
- Process environment
- Files and directories
- Input and output primitives
- Device- and class-specific functions
- Language-specific services for C programming
- System databases
- Data interface format

The additional system services allow users to perform operations such as process creation and execution, file system access, and I/O device management.

POSIX for OpenVMS also implements draft standard 1003.1a, the system interface extension (as listed in Table 2–1).

4.4.3 POSIX Commands and Utilities

POSIX for OpenVMS supports the POSIX 1003.2 standard, which includes an interactive interface (described in Section 5.3) and a callable interface to POSIX shell and utility services. The interactive interface to POSIX for OpenVMS is a set of commands and utilities similar to UNIX commands and utilities with many of the same functions and features of the Korn shell. These commands and utilities include those that provide functions similar to DCL (see Section 5.5.3) in addition to functions not available in the DCL environment. (The use of DCL commands is mentioned in Section 4.5 and described in Section 5.2.1.)

POSIX for OpenVMS supports a number of utilities and features based on UNIX that are unavailable in the basic DCL environment, including:

- Pipes, which allow the output of one command to become the input to the subsequent command. (In DCL, this process often requires the use of a temporary output file.)
- Complex utilities as listed in Table 4–7.
- Revision Control System: A series of utilities similar in function to CMS on OpenVMS.
- Utilities that facilitate the building of existing UNIX applications that conform to POSIX on an POSIX for OpenVMS system and provide the interface between the UNIX program development environment and the tools available in the OpenVMS environment (C compiler, OpenVMS Linker, OpenVMS Debugger):
  - The ar utility: Used to create and maintain libraries.

  - The ln utility: Used to create links between files and directories, allowing multiple routes of access.
  - The make utility: Used to build an application (similar to MMS in the OpenVMS environment).
- The vi text editor, which is a display-oriented editor familiar to UNIX users, and ed and ex editors.

Table 4–7 Complex Utilities Supported by POSIX for OpenVMS

Utility	Description
sed, grep	Batch-stream editors useful for editing extremely large files or making a change to a group of files according to a script of editing commands.
awk	A batch-stream editor that executes specified actions based on test patterns, using its own syntax (similar in many respects to C syntax); used interactively or in batch mode.
bc, dc†	Arbitrary precision arithmetic calculation facilities that use a C-like syntax; used interactively or in batch mode.
lex	A lexical analyzer generator that reads a description of lexical syntax from input files and generates C language code that performs lexical analysis; this C code can be used by the yacc utility.
mail, mailx	Utilities that support mail services.
yacc (yet another compiler compiler)	A tool for writing compilers and command interpreters that parse input according to specific grammar rules, creating tables that are used with C code to constitute a parser that will recognize the grammar.
c89	The POSIX interface to the compiler and linker (analogous to the cc command in UNIX).

†dc is used by UNIX like systems but does not conform to POSIX standards or draft standards.

A set of callable interfaces can be used to execute shell commands, compile and execute regular expressions, and perform pattern matching. Shell programming uses shell scripts to execute complex sequences of shell commands.

4.4.4 POSIX Real-Time Functions

POSIX for OpenVMS implements the POSIX 1003.1b (previously 1003.4) standard, which defines a set of real-time functions. For applications that have real-time computing requirements, these extensions provide support for such functions as enhanced interprocess communication, scheduling and memory management control, and asynchronous I/O operations. The following categories of real-time functions are supported:

- Semaphores, synchronization mechanisms to control access to systemwide resources
- Interprocess communication between multiple processes, support in the areas of event notification, message queues, and shared memory
- Asynchronous event notification, a way of passing data within an application
- Clocks and timers that let the application set the systemwide clock

4.4.5 POSIX XPG Support

POSIX for OpenVMS supports the BASE specifications described in XPG3 (the X/Open Portability Guide Issue 3), implementing the minimum set of components required to create the XPG3 Common Applications Environment. This set of components consists of the internationalized system calls and libraries, commands and utilities, and the C language.

XPG3 defines the run-time behavior of the library routines and how this behavior is affected by the internationalization environment but does not specify the tools to create and maintain the environment itself. POSIX standard 1003.2 defines the following tools for the internationalization environment:

- localdef: Used to define the internationalization environment
- locale: Used to query any internationalization environment currently available on the system

XPG3 internationalization features supported by POSIX for OpenVMS allow users to develop applications that can be deployed (without recompiling) in multiple cultures or countries in such a way that users can experience their own language and cultural context. The application developer can use these features to ensure that messages from the application are generated in the local language context.

POSIX for OpenVMS VAX and OpenVMS Alpha also supplies the XPG3 curses library, which provides a series of callable functions for cursor control optimization.

In addition, POSIX for OpenVMS has received XPG4 BASE profile branding for the following software components:

- System calls and headers
- Wide character functions (which support multibyte characters for Asian languages)
- Command and utilities required for XPG4 BASE profile branding
- The C language, as defined by XPG4

For additional information about POSIX for OpenVMS programming interfaces, refer to the POSIX for OpenVMS documentation set.

4.5 Programming User Interfaces

User interfaces to the OpenVMS VAX and OpenVMS Alpha operating systems include the DCL interface (see Section 5.2.1) and the optional DECwindows Motif graphical user interface (see Section 5.2). Another user interface is through electronic forms (see Section 5.4.1).

DCL commands can be used to invoke program development software (compilers, editors, linkers) and to run and control execution of programs. DCL command procedures can be used to perform repetitious operations in software development.

The Command Definition utility (CDU) enables application developers to create DCL-level commands with a syntax similar to OpenVMS DCL commands. Using CDU, the developer can create applications with user interfaces similar to those of operating system applications. The Message utility permits an application developer to create application messages to supplement the system messages supplied by the OpenVMS operating system.

The DECwindows Motif software provides a consistent user interface for developing software applications and includes an extensive set of programming libraries and tools. DECwindows Motif supports both the OSF/Motif standards-based graphical user interface and the X user interface (XUI) in a single run-time and development environment. DECwindows Motif for OpenVMS Alpha requires a DECwindows X11 display server (device driver and fonts) that supports the portable compiled format (PCF), permitting use of vendor-independent fonts.

An applications programmer can use the following DECwindows Motif software to construct a graphical user interface:

- A user interface toolkit composed of graphical user interface objects (widgets and gadgets); widgets provide advanced programming capabilities that permit users to create graphic applications; gadgets, similar to widgets, require less memory to create labels, buttons, and separators
- A user interface language to describe visual aspects of objects (menus, labels, forms) and to specify changes resulting from user interaction
- The OSF/Motif Window Manager, which allows users to customize the interface

The DECwindows Motif programming libraries provided include:

- Standard X Window System libraries such as Xlib and the intrinsics
- Libraries needed to support the current base of XUI applications
- OSF/Motif toolkit support for developing applications using the Motif user interface style
- Digital libraries that give users capabilities beyond the standards

Also included in the DECwindows Motif environment are DEClinks services for creating, managing, and traversing informational links between different application-specific data. DEClinks services, with the DEClinks Manager application, help organize information into a hyperinformation environment (see Section 5.2.2). DEClinks Developer's Tools provide a development environment for creating, modifying, and maintaining hyperapplications.

4.6 Developing Real-Time Applications

The VAXELN Toolkit is a set of tools that can be used on VAX systems running OpenVMS to develop efficient real-time applications (for example, real-time applications for process control, simulation, or high-speed data acquisition). VAXELN real-time applications are run on rtVAX computers. The VAXELN real-time operating environment optimizes dedicated real-time systems for predictability and fast response time. VAXELN transparently supports open standards such as IEEE 1003.1 and IEEE 1003.1b (previously 1003.4). VAXELN operates in TCP/IP and DECnet local and wide area networks.

4.7 Digital Software Development Tools

Digital supplies optional software development tools for the OpenVMS environment, such as DECset. DECset is a set of tools that support software coding, testing, and maintenance of applications and data. These tools can be used individually or as part of the optional Digital software development environment.

Use of software development tools on the OpenVMS system in multivendor distributed environments is described in Section 6.3.5.

4.8 Managing Data

The basic OpenVMS tool for transparent, intuitive management of data is the Record Management Services (RMS) subsystem. RMS is a collection of routines that give programmers a device-independent method for storing, retrieving, and modifying data for their application. RMS also provides extensive protection and reliability features to ensure data integrity.

RMS is a higher level interface to the file system and OpenVMS I/O subsystem. It is used by all products that run on OpenVMS VAX and OpenVMS Alpha for file and record operations. A subset of RMS services permit network file operations that are generally transparent to the user.

On OpenVMS Alpha systems, RMS supports I/O operations to and from 64-bit addressable space.

4.8.1 RMS Files and Records

RMS supports a variety of file organizations, record formats, and record-access modes. RMS supports sequential, relative, and indexed disk file organizations, and fixed- and variable-length records. It supports a number of record-access modes: sequential, by key value, by relative record number, or by record file address. RMS is designed primarily for mass storage devices (disks and tapes), but also supports unit-record devices such as terminals or printers.

RMS routines assist user programs in processing and managing files and their contents. RMS routines perform these services for application programs:

- Creating new files, accessing existing files, extending disk space for files, closing files, and obtaining file characteristics
- Getting, locating, inserting, updating, and deleting records in files

RMS promotes safe and efficient file sharing by providing multiple access modes, automatic record locking when applicable, and optional buffer sharing by multiple processes.

RMS files are also used in the POSIX for OpenVMS environment. Files created by POSIX applications on OpenVMS are binary stream RMS files. POSIX developers can handle files in alternative ways. In the OpenVMS environment, the POSIX developer can create files using OpenVMS naming techniques and then use DCL to access all files generated by OpenVMS or POSIX. In the POSIX for OpenVMS environment, to obtain full POSIX file compliance, a container file system is required as part of the POSIX file name space. A container file system is an RMS directory with an extra file to map between POSIX file names and RMS file names. The container file system is useful in implementing a file intended to be used on multiple networked platforms, including NFS file systems.

For a description of POSIX files and directories on OpenVMS systems, see Section 5.5.2.

4.8.2 RMS Utilities

RMS file utilities allow users to analyze the internal structure of an RMS file and to determine the most appropriate set of parameters to tune an RMS file. RMS utilities can also be used to create, efficiently load, and reclaim space in an RMS file.

RMS file maintenance utilities include the following:

- Analyze/RMS_File utility
- File Definition Language utilities (Create/FDL and Edit/FDL)
- Convert and Convert/Reclaim utilities

The Analyze/RMS_File utility permits the programmer to analyze the internal structure of an OpenVMS RMS file and generate a report on its structure and use, as well as interactively explore the file's structure. The utility can generate an FDL file from an RMS file for use with the Edit/FDL utility to optimize the data file.

File Definition Language (FDL) is a special-purpose language for specifying file characteristics; it is useful with higher level languages or for ensuring that files are properly tuned. FDL makes use of RMS control blocks: the file access block (FAB), the record access block (RAB), and the extended attribute block (XAB).

The Edit/FDL utility creates a new FDL file according to user specifications. The Create/FDL utility uses the specifications of an existing FDL file to create a new empty data file.

The Convert utility can be used to copy records from one file to another, while changing the record format and file organization, and to append records to an existing file. The Convert/Reclaim utility reclaims empty bucket space in an indexed file to allow new records to be written to it.

5
User Interfaces to the OpenVMS System

The OpenVMS system offers users consistent, easy-to-use interfaces: a natural language command interface and an optional standards-based graphical user interface, as well as specialized forms-based interfaces. People throughout an organization or activity can use these consistent interfaces to access authorized information and resources anywhere in the OpenVMS software environment.

Users can access the OpenVMS operating system from a wide variety of devices, ranging from desktop devices to terminals connected to large computer complexes. Depending on the configuration, OpenVMS users can share in the full capabilities of OpenVMS VAX and OpenVMS Alpha systems and VMScluster systems and can access resources on other computers throughout a worldwide multivendor network.

OpenVMS support for open standards permits development of user-portable applications. A user-portable application provides the user with a consistent interface when the application is run on any system (including systems from other vendors) that conforms to the same open standards. The user experiences the same environment from system to system, without the requirement for retraining. OpenVMS support for open, portable software capabilities is discussed in Section 2.1.4.

This chapter describes user access to OpenVMS systems, the different user interfaces, and the various software environments users can access. The chapter also covers OpenVMS support for such general user activities as file handling and electronic mail.

5.1 OpenVMS Operating System Access

Access to the OpenVMS VAX system or the OpenVMS Alpha system is controlled by means of user accounts. To log in and gain access to the OpenVMS system, the user supplies the user name and password specified in the appropriate user account. Information about each user account is maintained in the user authorization file (UAF), which can be modified by the OpenVMS system manager (see Section 3.2.2).

The system manager can determine the user's needs and set up an account that controls the user's access to system resources. In establishing an account for an individual user, the system manager assigns a unique user name, password, and user identification code (UIC) for the account, and specifies the system privileges and resource quotas associated with the account. Examples of protection mechanisms the system manager can use to control access to system objects (such as files) include the following:

- UIC that identifies the user as a member of a group that can share specific data

- Access control list (ACL) that permits specific users to be included or excluded from accessing a file or achieving certain kinds of access (for example, read or write access)

For additional information about user accounts and other protection mechanisms employed to control user access to the system, see Section 3.3.2.

Some users can access a turnkey or captive account without supplying a user name or password. For example, turnkey accounts can be established for special environments such as a factory floor, in which users perform routine tasks and do not need to submit commands to the operating system. Special limited-access accounts are described in Section 3.3.2.

If a user is logged in to an OpenVMS system connected to a network and has access to an account on another node (which need not be an OpenVMS node) in the network, the user can log in to that account from the local node and use the facilities of that remote node while remaining physically connected to the local node. For example, an OpenVMS user can enter the SET HOST command to access a remote node and then log in to the remote node.

Optionally, the user can enter different user environments when logging in to the OpenVMS system. The particular environment is established by the user account and by the way in which the user accesses the system. Possible user environments include:

- The OpenVMS environment (if neither the user nor the system manager specifies an alternative login environment)
- The POSIX for OpenVMS environment (if either the user or the system manager explicitly specifies it)
- A customized forms-based environment, such as the ALL–IN–1 integrated office environment (if either the user or the system manager specifies it)

The POSIX for OpenVMS environment supported by the OpenVMS system is similar to a UNIX environment (see Section 2.1.3.1). POSIX is designed to enable users and applications that comply with POSIX standards to move between systems. The POSIX user environment on an OpenVMS system is described in Section 5.3.

Optional user interfaces based on forms are described in Section 5.4.

5.2 OpenVMS User Environment

When a user logs in to the OpenVMS operating system, the system creates an environment from which the user can enter commands and run programs. The environment is called a process. The characteristics of the process are specified in the user authorization file (UAF) set up for the user. Within the process, the system executes programs (called images or executable images) one at a time. An image is a file that contains machine-readable instructions and data. The system and the user can both supply image files.

Users can access OpenVMS through the DIGITAL Command Language (DCL) interface supplied with the system (described in Section 5.2.1) or the optional DECwindows Motif graphical windowing interface (described in Section 5.2.2).

When a user logs in to the OpenVMS system, two DCL command procedures are automatically executed: a system login command procedure that the system manager sets up and a personal login command procedure that either the system manager or the user can set up. Editing the personal login command procedure

permits the user to customize the OpenVMS computing environment. (DCL command procedures are described in Section 5.2.1.1.)

5.2.1 DCL Command Language Functions

DCL provides the user with a consistent user interface. The command language is a set of English-like instructions that tell the OpenVMS system to perform specific operations. The DCL command takes the form of a command name followed by parameters and qualifiers, as appropriate. OpenVMS users can use DCL to conduct a dialogue with the system or to initiate system utilities or user programs.

Users on character-cell terminals can enter DCL commands interactively. OpenVMS users on workstations can enter DCL commands using the DECterm window in the DECwindows Motif interface. OpenVMS users can also include DCL commands in command procedures, invoking the command procedures interactively or submitting them in batch queues for deferred execution. DECwindows users can use the point and click technique to execute DCL commands, using the FileView window.

The range of tasks that can be performed using DCL commands is extensive. For example, DCL commands permit the user to:

- Get information about the system
- Modify the work environment
- Work with files
- Work with devices (such as disks and magnetic tape)
- Develop and execute programs
- Provide security and ensure that resources are used efficiently

Table 5–1 lists the types of computing tasks and examples of the DCL commands used to perform the tasks. Most DCL commands are supported on OpenVMS VAX and OpenVMS Alpha systems; some DCL qualifiers and options are system specific. DCL commands are described in alphabetic order in the *OpenVMS DCL Dictionary*.

Table 5–1 Types of Tasks Performed by Commonly Used DCL Commands

Types of Tasks	Examples of DCL Commands
General session control and environmental control	HELP, SHOW, SET, ASSIGN, DEFINE, DEASSIGN, PHONE, MAIL, LOGOUT, REQUEST
Volume and device resource control	MOUNT, INITIALIZE, DISMOUNT, ALLOCATE, DEALLOCATE
Program development and execution control	LIBRARY, LINK, RUN, DEBUG, SPAWN, STOP, SUBMIT, SYNCHRONIZE, WAIT, CANCEL
File manipulation control	DIRECTORY, CREATE, EDIT, DELETE, PURGE, RENAME, COPY, APPEND, DIFFERENCES, SORT, OPEN, CLOSE, READ, WRITE, PRINT, TYPE, DUMP, ANALYZE

Users can also use special DCL commands for other purposes, such as customizing the OpenVMS user environment to make it more productive. Certain DCL commands define short, meaningful logical names for commonly used files, directories, and devices. The system also creates a set of systemwide logical names when the user logs in. Other commands create symbols that

represent DCL commands, logical values, or numeric or character values that can be combined into expressions to manipulate the values the symbols represent. Lexical functions permit the user to obtain information from the system about system processes, queues, and user functions. Lexical functions are essentially calls to OpenVMS system services.

For detailed information about using DCL, see the *OpenVMS User's Manual*.

5.2.1.1 DCL Command Procedures

Command-level programming permits the user to create DCL command procedures using a text editor or the DCL command CREATE. A command procedure comprises DCL commands to be executed, data lines that are used by those commands, and optional comment lines. Command procedures can be started interactively or by another command procedure. Command procedures can also be run as batch jobs in a batch queue. Using batch mode permits submission of work for execution at a specified time (for example, during off hours). An advantage of batch mode is the ability to restart the procedure if there is a system failure: DCL provides a /RESTART option that causes the system to restart a job if the system crashes before it is completed.

DCL command procedures provide a full programming language for operator and user interfaces. Command procedures and lexical functions can be used to automate routine system management tasks.

Within command procedures executed at the local node, the user can use DCL commands to open and close local and remote files, and read and write records to those files. The user can also submit DCL command procedures residing on remote nodes for execution as batch jobs on those nodes.

5.2.1.2 OpenVMS Help System

DCL and OpenVMS utilities have a consistent integrated help subsystem to provide operational information on all aspects of system operations. The DCL HELP command permits the user to obtain information about using the system, available DCL commands, and system utilities. The online help system includes summary operational information on all aspects of OpenVMS system operation.

In addition, the Help Message utility (MSGHLP) allows users to access instant online descriptions of system messages at a character-cell terminal or a DECterm window in a DECwindows Motif interface. Operating through the DCL interface, this utility can provide information about messages supplied by OpenVMS, by optional products running on OpenVMS, and by the customer, if desired. For information about the Help Message utility, refer to *OpenVMS System Messages: Companion Guide for Help Message Users*.

5.2.2 OpenVMS DECwindows Motif Interface

OpenVMS users on VAX or Alpha workstations can access the optional product DECwindows Motif, an open graphical windowing interface enhanced by a set of integrated desktop applications. The DECwindows Motif graphical user interface complies with the Motif specification developed by the Open Software Foundation (OSF) (see Section 2.1.3.3). OSF/Motif specifies both a graphical user interface and an application programming interface. DECwindows Motif is based on the Massachusetts Institute of Technology specification for the X Window System, the de facto standard for distributed windowing systems.

DECwindows Motif supports an interactive, network-based client/server environment. DECwindows applications can run on a client node and display output on an OpenVMS server node. In other words, an application running on a remote machine can display on the user's OpenVMS desktop device, and the user can interact with the application as though it were running locally. Clients and servers communicate with each other by sharing local memory or over DECnet or TCP/IP network connections.

The DECwindows Motif end-user environment involves point-and-click techniques for interacting with the OpenVMS operating system. The user can divide the workstation screen into multiple windows, each representing a different application. DECwindows applications automate many basic tasks (for example, Mail automates sending and receiving interoffice mail).

A workstation mouse or other pointing device permits the user to point to and select text and objects on the screen. Users interact with DECwindows by finding an object on the screen that represents a task, selecting that object, and completing that task. Typical interface objects include:

- Menus, containing items that let the user tell DECwindows what the user wants to do or wants to work with (for example, the Help menu from an application)
- Icons, representing currently running applications that need to remain available but are too large for the screen
- Buttons that affect the positioning of windows on the screen (for example, the minimize button that lets the user shrink a window to an icon and store it as an icon on the screen)
- Dialog boxes that permit the user to supply additional information to the system

Figure 5–1 shows the main DECwindows Motif components. These components include:

- Session Manager: Appears at the start of every DECwindows session and permits the user to control the DECwindows session, customize the DECwindows environment, and run applications.
- FileView: Gives the user a graphical access to DECwindows applications and provides commands for working with files and directories.
- Workspace: The screen on the workstation and the background of the DECwindows environment. All windows and objects appear on the Workspace.

Figure 5–1 DECwindows Motif User Interface

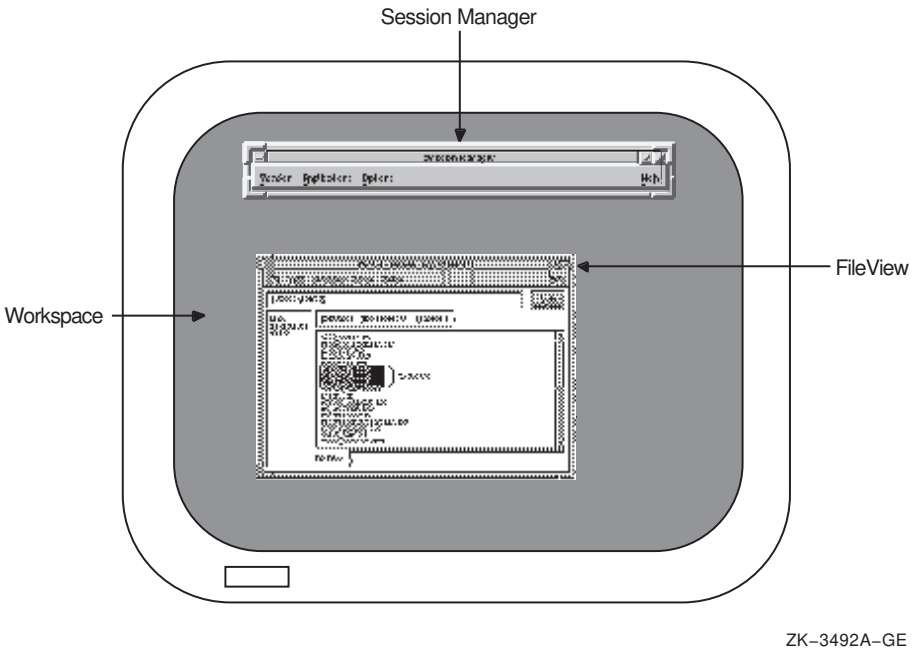


Table 5–2 briefly describes DECwindows Motif applications.

Table 5–2 Commonly Used DECwindows Motif Applications

Application	Function
Bookreader	Displays an online documentation reader on the workstation screen
Calculator	Performs mathematical operations
Calendar	Keeps track of scheduled appointments and assists in planning the user's time
Cardfiler	Organizes information with index cards and card files
CDA Viewer	Displays the contents of many different types of files on the workstation screen (for example, PostScript files)
Clock	Displays the time of day (in both analog and digital format) and the date on the workstation screen with alarm features
DECsound	Records, saves, stores, and plays back audio images
DECterm	Creates a window that emulates a VT300-series terminal
DEClinks Manager	Controls the hyperinformation environment
Mail	Lets the user exchange messages with other computer users
Paint	Creates a simple picture, an illustration, or a map
Print Screen	Takes a snapshot of the entire screen or a portion of it and prints the file containing the snapshot or saves it
Puzzle	Displays a video version of a number puzzle that contains squares to arrange in ascending order

DECsound is an easy-to-use application that lets the user record and play back audio images on a subset of workstation models. DECsound includes sample sounds and provides support for mailing audio messages across the network. It

also enables audio messages to be included in compound documents that can be played by the CDA Viewer.

The DECterm terminal emulator provides a traditional character-cell interface, which permits users to enter DCL commmands or use any other command-line interface.

DEClinks, built on an open, object-oriented data model, is a set of services for creating, managing, and traversing informational links between different multimedia data, such as electronic mail, online manuals, pictures, design notes, and slide presentations. DEClinks can link video, audio, image-processing, and electronic publishing data. This linked environment is called the hyperinformation environment. Hyperapplications permit application end users to link information objects through a Link menu interface. DEClinks services, with the DEClinks Manager application, help organize information by allowing users to focus on the information rather than on where the information comes from and where it is stored.

DECwindows Motif provides a common look and feel across various software environments and tools. The application programmer can construct the user interface using OSF/Motif software tools. The DECwindows tools package for Motif is described in Section 4.5.

5.3 POSIX Environment on an OpenVMS System

A user logging in to an OpenVMS VAX or OpenVMS Alpha system that includes POSIX for OpenVMS can invoke the POSIX environment, a multiuser timesharing environment supported on character-cell terminals. The environment complies with POSIX standard 1003.2, which supports terminal users in a consistent manner across all conforming systems. Users in the POSIX environment need to be familiar with the style of interaction with UNIX systems. Typical users would be program developers, engineers, or general-purpose timesharing users interested in portable applications that are similar to UNIX applications.

The POSIX environment differs from the OpenVMS environment. Commands are limited to those that the POSIX environment understands. Some POSIX commands have the same name as DCL commands, but the functions are different. The command-line interface includes UNIX features such as scripts and complex utilities (see Section 4.4.3).

For detailed information about the POSIX for OpenVMS user environment, refer to the POSIX for OpenVMS documentation set.

When logging in to a POSIX for OpenVMS system, the user can log in directly to the DCL environment or to the POSIX for OpenVMS shell environment. The POSIX for OpenVMS shell is a command language interpreter (CLI) equivalent to the OpenVMS DCL interpreter. The user can instruct the shell, either interactively or within a POSIX for OpenVMS command file (called a shell script), to perform tasks such as calling utilities.

Logging in directly to the POSIX shell environment requires the addition of the qualifier /CLI=POSIX$CLI to the user name. Alternatively, the user can log in to the OpenVMS DCL environment and invoke the POSIX for OpenVMS shell.

On an OpenVMS system that includes POSIX for OpenVMS, the user can move back and forth between the OpenVMS and POSIX for OpenVMS environments.

5.4 Forms-Based User Environments

A user logged in to an OpenVMS system may interact with application software by responding to electronic forms. An electronic form is a collection of fields and background text displayed on any type of display device. Forms are used with a variety of applications to enhance the gathering and display of information.

The following sections describe the Digital DECforms software product and the forms-based integrated office environment, ALL–IN–1.

5.4.1 DECforms Interface

Some applications that OpenVMS users can access provide user-friendly forms interfaces. The OpenVMS forms software product, DECforms, allows a single application to support multiple types of users with interfaces tailored to their needs, including interfaces in multilingual environments. DECforms supports the full range of OpenVMS terminals and compatible terminal emulators on workstations and personal computers.

DECforms is the OpenVMS implementation of the proposed ANSI ISO standard for a Forms Interface Management System (FIMS), which standardizes the interface between an application and the forms it uses. DECforms offers a subset of the full FIMS functionality, with extensions tailored for the OpenVMS environment. DECforms embodies the fundamental principles underlying the FIMS model:

- Separation of form and function
- Efficient distribution of forms processing
- Ease of use
- Flexibility of user interface control
- Programming language independence

DECforms combines the capabilities of previous Digital forms systems (VAX FMS and VAX TDMS) and adds new features. When DECforms is integrated with the ACMS product, it serves as the user interface into database products (such as Oracle Rdb and Oracle CODASYL DBMS) that run on OpenVMS VAX and OpenVMS Alpha systems. The integration of DECforms with ACMS provides powerful forms-processing capabilities in transaction-processing environments (see Section 7.2.2). A single DECforms run-time process can control multiple terminals simultaneously. Using optional software, DECforms can be distributed to remote CPUs, bringing forms processing as close to the end user as possible. DECforms services are also included in middleware integrated products.

5.4.2 ALL–IN–1 Office Systems Environment

OpenVMS supports an optional, specialized user environment: the forms-based, menu-driven, integrated ALL–IN–1 system. ALL–IN–1 links office applications together and includes a facility for integrating other business-oriented applications. ALL–IN–1, which is easy to use, controls user activities.

When a user who has an ALL–IN–1 user account logs in to an OpenVMS system, the user can then log in to the ALL–IN–1 system. OpenVMS users on workstations can access the ALL–IN–1 system through a dynamic graphical user interface provided by the ALL–IN–1 Services for DECwindows.

ALL–IN–1 communicates with users through forms displayed on the terminal screen. The first form displayed is the Main menu, followed by additional options, each representing a subsystem or specific group of office tasks.

ALL–IN–1 allows users to transport or receive information from other systems through the electronic messaging facility linked to the network. User documents are stored in folders in the user's file cabinet where they are accessed by the office applications. Users can create and process documents using either the EDT or WPS editor or the WPS–PLUS full-function word processor.

ALL–IN–1 is a customizable software product. Using a client/server model to provide the core services that all office workers need, ALL–IN–1 adds advanced capabilities for workgroup computing, enterprise communications, and information management.

5.5 Information Handling on the OpenVMS System

End users on the OpenVMS operating system normally work with information stored in files. OpenVMS supports files that can be used in either the OpenVMS environment or the POSIX environment. In addition, POSIX for OpenVMS files can be ported to other systems that conform to POSIX standards.

In the OpenVMS environment, users can enter DCL commands to access and manipulate files and to sort and merge records in files. Software tools are available to create, edit, and process text files. The DECwindows Motif interface also supports file handling through the FileView program, which is a graphical interface to OpenVMS file management (see Section 5.2.2).

5.5.1 OpenVMS Files and Directories

An OpenVMS file can consist of text the user enters and manipulates or machine-readable data that the machine understands. Creating a memo is creating a file; sending an electronic mail message is sending a file. Running a program involves loading an image file into the system and executing the instructions contained in that file.

OpenVMS files are listed in directories. Each directory is a special file that contains the names and locations of files. Every OpenVMS user account has an associated disk directory containing user files. This default or login directory can include many levels of subdirectories, arranged in a hierarchical directory structure.

The full file specification for an OpenVMS file describes the access path the OpenVMS system uses to locate and identify a file. The specification identifies the location at which the file resides: the network node name, device name of the directory disk volume, directory name, and file name, type, and version number of the file. The file type identifies the structure or type of data in the file. (If the node name and directory name are not specified, the defaults are the user's current node and directory name.) Wildcard characters can be used to manipulate large numbers of files without naming them individually (for example, the asterisk can be used to indicate all values of certain fields, or parts of fields, in a file specification).

5.5.2 POSIX Files and Directories

Users in the POSIX for OpenVMS environment can choose between using OpenVMS files or POSIX files (similar to UNIX files) for their applications. Users of POSIX for OpenVMS can consider the file system as containing two parts:

- The OpenVMS file system in the context of the POSIX for OpenVMS environment. OpenVMS files can be used in the POSIX environment for interoperability with other components of the OpenVMS operating system, but must be referred to by POSIX pathnames.
- The container file system (see Section 4.8.1). The POSIX for OpenVMS container file system permits a file name on an OpenVMS system to be translated to a file that fully supports the POSIX standards and is portable to other systems that conform to POSIX standards.

POSIX provides the translate utility, which enables translation between OpenVMS and POSIX of pathnames and file names.

POSIX files follow rules similar to the UNIX environment. POSIX files are referred to by pathnames that indicate the directory path to the file, including the file name itself (no device name is specified). Subdirectories are preceded by the slash character.

The OpenVMS file system and the POSIX container file system differ in file specifications, file structures, file protection, use of special files, links between files, and symbolic links.

Software development involving POSIX files is discussed in Section 4.8.1. For additional information about using files in the POSIX for OpenVMS environment, refer to the *POSIX for OpenVMS Guide to Programming*.

5.5.3 OpenVMS File Manipulation

In the OpenVMS environment, users can perform file operations by specifying DCL commands. Some of the commands invoke OpenVMS utilities that perform the file operation. File operations performed using DCL commands are summarized in Table 5–3.

Table 5–3 File Operations Performed Using OpenVMS Utilities and DCL Commands

Utility or DCL Command	File Operation
BACKUP utility	Saves and restores user data
CONVERT utility	Maintains optimal ISAM file performance, copies records to files of different organization, reformats indexed files
COPY command	Moves files around
CREATE command	Creates files or directories
DELETE/ERASE command	Erases a file and removes it from a directory
DIRECTORY command	Displays the contents of a current directory
DUMP command	Displays actual file contents to facilitate application debugging

(continued on next page)

Table 5–3 (Cont.) File Operations Performed Using OpenVMS Utilities and DCL Commands

Utility or DCL Command	File Operation
EDIT command	Permits creation of a new file or viewing and changing of the contents of a text file
PRINT command	Sends a specified file to a printer for printing
PURGE command	Facilitates disk space reclamation by deleting old versions of a file
RECOVER command	Applies RMS journals to recover RMS files
RENAME command	Changes the name of a specified file
SET FILE command	Manipulates file characteristics
SORT/MERGE utility	Sorts records in a file and combines previously sorted files into an output file
TYPE command	Displays the contents of a specified file

Users do not normally manipulate the records that comprise a data file, except to sort or merge records in a file. The OpenVMS Sort/Merge utility is used to manage the records. The Sort utility sorts records from as many as 10 input files into an output file, based on a sequence of user-selected keys in the input files. The user can define key fields on which to organize the files in alphabetic or numeric order, in ascending (low to high) or descending order. The Merge utility combines up to 10 previously sorted files according to a user-selected key and generates an ordered output file.

On OpenVMS Alpha systems, a high-performance sort takes advantage of the Alpha architecture to provide better performance for most sort and merge operations. The high-performance sort uses the same command line interface as the Sort/Merge utility. The high-performance sort can sort records from as many as 12 input files. Many existing sort and merge operations can use the high-performance sort without modification.

OpenVMS users can perform DECnet network file operations as a natural extension of the I/O operations performed on their systems. Most DCL commands permit the user to access files on remote systems in the same way as on the local system. Only a node name need be added to the file specification for network operations. Certain commands that invoke processing on a specific system require the /REMOTE qualifier (for example, PRINT/REMOTE and SUBMIT /REMOTE).

The DCL command EXCHANGE/NETWORK allows the transfer of files to or from heterogeneous operating systems that do not support OpenVMS file organizations, over DECnet communications links. For example, users can transfer files between MS–DOS or UNIX systems and OpenVMS systems, with the option of modifying file and record attributes.

Some DCL commands have qualifiers that make frequently used TCP/IP functions easier to access in a more integrated manner. These commands include:

- COPY/FTP and COPY/RCP
- DIRECTORY/FTP
- SET HOST/RLOGIN, SET HOST/TELNET, and SET HOST/TN3270

For these commands to work, a separate TCP/IP product that supports these commands (for example, Digital TCP/IP Services for OpenVMS) must be installed on the OpenVMS system.

In the POSIX for OpenVMS environment, POSIX interactive users use POSIX 1003.2 commands and utilities to manipulate files. Some POSIX commands perform the same function as DCL commands, but have different names. (For example, the POSIX command ls is equivalent to the DCL command DIRECTORY, and cd is equivalent to the DCL command SET DEFAULT.) Conversely, some POSIX commands have the same name as DCL commands, but the functions are different. (For example, the DCL command TYPE displays a file on the terminal. In a POSIX for OpenVMS environment, the command type displays the structure or type and pathname of a command, and either the more or cat utility displays a file on the terminal.)

5.5.4 Text File Editing and Processing

OpenVMS provides several editors for use in editing text files. Text editing is the process of creating and maintaining character-oriented files. Editors can also be used to create and modify source files for programming languages or text formatters (such as VAX DOCUMENT). Two of the most commonly used editors are EVE and EDT:

- The Extensible Versatile Editor (EVE) allows the user to insert, change, and delete text quickly. EVE is written in the DEC Text Processing Utility (DECTPU) language, described in Section 4.2.1. EVE is a full-screen editor that allows users to scroll through text on a terminal screen. EVE provides an EDT-style keypad.
- EDT permits the user to enter and manipulate text and programs and perform line and character editing. EDT is an easy-to-learn editor, with a help system and a journaling capability to protect against loss of edits. EDT provides screen editing using the keypad on VT-series terminals.

Another text editor that runs on OpenVMS is the Text Editor and Corrector (TECO), which is used to edit ASCII files.

POSIX for OpenVMS supports the vi utility, a display-oriented interactive text editor that includes the line editor, ex. It also supports the ed utility, a line-oriented text editor. Both utilities are defined in POSIX 1003.2 and in the XPG3 and XPG4 BASE specifications.

Text formatters are used to prepare documents, processing source files into formatted text and creating tables of contents and indexes. Text processors available to OpenVMS users include:

- The DIGITAL Standard Runoff (DSR) facility, a basic text formatter supplied with the OpenVMS system
- VAX DOCUMENT, a text formatter that provides the tools for automated book publishing and the capability to produce online documentation for the Bookreader application

Optional products provide for text processing and multimedia document preparation and control on OpenVMS systems; these services are also available with middleware products (see Section 6.3.1).

5.5.5 Electronic Mail

OpenVMS users can communicate with users on their own system and on other systems electronically. The OpenVMS system's communication capabilities are designed to operate over a network as easily as on a single system.

The standard electronic mail facility for OpenVMS users is the OpenVMS Mail utility, which permits a user to send messages to, and receive messages from, any other user on the same system or to users on other systems on the same network. Once a user is logged in to the OpenVMS system, Mail displays a message on the terminal screen with a notification of any incoming mail. A user can invoke the Mail utility and, at the MAIL> prompt, send or read mail, or perform other operations such as forwarding or replying to mail, deleting messages, or extracting messages to create text files. A user can also organize a mail file by filing mail in folders and displaying the messages contained in each folder. For more information about using the OpenVMS Mail utility, refer to the *OpenVMS User's Manual*.

The Mail utility provides support for TCP/IP by incorporating features that are standard on the Internet. Mail has been enhanced to provide support for:

- Parsing of an RFC 822 (username@node) format address with the following limits on the RFC 822 format: the address must be on one line (with no CR character), and an address that contains any space characters must be enclosed in quotation marks.
- Signature files that can be appended to mail messages automatically (for example, your name, address, and telephone number)
- Standard screen functions (scrolling) for reading messages and reviewing directory listings

Additionally, the DECwindows Motif desktop environment supports the use of the DECwindows Mail application (as described in Section 5.2.2).

OpenVMS users in the ALL–IN–1 environment can send mail electronically across a heterogeneous network using messaging services based on middleware. MailWorks for OpenVMS provides access to the full range of Digital multivendor connectivity services by means of the MAILbus facility. The MAILbus family of products links multivendor electronic mail systems and messaging applications into an enterprisewide electronic messaging system.

5.5.6 Electronic Conferencing and Text Retrieval Facilities

Other optional communications software permits the OpenVMS user to access the DEC Notes electronic conferencing facility and the Digital VTX text retrieval facility. DEC Notes supports group conferencing, organizing online discussions into topic-and-reply formats that both encourage and keep a record of team communication. DEC Notes can be accessed from a broad variety of desktop devices. VTX enables users on various desktop devices to access online information (such as documents intended for a wide audience), across the network, using client/server computing. Users find the information they need by responding to a series of menus.

Part III

Open Distributed Computing Environments

This part of the manual describes computing environments in which OpenVMS software can participate.

Chapter 6 describes integrating OpenVMS systems with other systems in distributed heterogeneous networks and using middleware to support multivendor integration. The chapter explains how OpenVMS systems and VMScluster systems are used in client/server configurations.

Chapter 7 presents high-integrity features of OpenVMS systems in distributed production system environments. It covers optional commercial-strength software that runs on OpenVMS, providing dependable transaction-processing capabilities in open environments.

6

OpenVMS Systems in Distributed Environments

OpenVMS software can be used in an OpenVMS standalone or VMScluster system configuration or as part of a complex distributed environment involving software from multiple vendors.

This chapter describes integrated environments in which OpenVMS operating systems running optional (layered) software are connected to other systems provided by Digital and other vendors. The chapter summarizes software functions available in environments specific to OpenVMS and additional functions useful in open, distributed environments. It describes software[1] running on OpenVMS that is relevant to distributed multivendor environments, such as networking products and middleware products that support multivendor application integration. The chapter also describes the participation of OpenVMS software in distributed client/server relationships, including the use of OpenVMS servers to provide services to clients on PCs and other desktop systems.

6.1 OpenVMS Functions Applicable to Distributed Environments

The OpenVMS system provides full computing capabilities to OpenVMS users in standalone, VMScluster, and networked configurations (that involve connections to other OpenVMS systems and to other systems provided by Digital and other vendors).

A standalone OpenVMS system or a VMScluster system that is not connected to other systems can support all basic software functions, including:

- OpenVMS base operating system functions (with built-in features such as help, Mail, and DECdtm along with integrated software such as VMSclusters, volume shadowing, and RMS Journaling)
- POSIX capability installed in the OpenVMS base operating system
- The full set of OpenVMS utilities
- Management of the OpenVMS system or VMScluster system
- Full OpenVMS security features
- Program development capabilities
- The ability to run most OpenVMS layered products and applications

These computing capabilities, available to OpenVMS users on every standalone OpenVMS or VMScluster system, are covered in Part II of this manual.

[1] Unless otherwise indicated, optional products mentioned in this manual run on OpenVMS Alpha systems as well as OpenVMS VAX systems. For further information about the availability of specific products, see your Digital sales representative.

OpenVMS systems connected in a distributed multivendor network perform the same functions as a standalone system while using the resources of the network. This type of environment can involve connection of different systems and integration of software from multiple vendors. Functions related to using OpenVMS software in open distributed environments include:

- Use of open standards-based software interfaces
- Use of portable software (such as portable applications that conform to POSIX standards)
- Use of networking connections
- Use of distributed applications
- Use of middleware for multivendor interoperability
- Integrated management of open systems and multivendor enterprises
- Sharing of resources with other systems
- Participation in multivendor client/server configurations

These software features permit OpenVMS systems to function cooperatively in a network with other kinds of computers: Digital and other vendor systems, from PCs to supercomputers.

Open interfaces, portable software features, and distributed computing capabilities provided by the OpenVMS system are summarized in Chapter 2. The following sections describe the open, distributed software environments that OpenVMS systems support.

6.2 OpenVMS Systems in Distributed Heterogeneous Network Environments

Digital networking supports comprehensive distributed computing environments that permit OpenVMS and other Digital systems to be connected to systems supplied by other vendors. The networking software allows connected systems to share resources and function as an integrated system, providing basically the same environment for users everywhere in the network.

OpenVMS supports the following networking services that can be integrated into a single multiprotocol network: the DECnet/OSI family of software and TCP/IP networking products (networking products are introduced in Section 1.3.4). DECnet/OSI software (described in detail in Section 6.2.1) permits connection to virtually all Digital systems and, in its most recent phase, to all systems that support the international standards for Open Systems Interconnection (OSI).

DECnet/OSI also supports links to TCP/IP transports through TCP/IP networking products, such as Digital TCP/IP Services for OpenVMS, running on the same system. The TCP/IP products permit connection to other systems running TCP/IP and to the Internet (see Section 6.2.2). DECnet/OSI also allows connections to other networking environments such as public packet-switching networks, PC LANs, and the IBM SNA network. Supporting these networking services are products suitable for multivendor environments, such as multiprotocol wide area network (WAN) routers (see Section 6.2.4).

DECnet, OSI, and TCP/IP networks can be integrated to operate as a single network, in such a way that the protocols used are transparent to applications and users (see Figure 1–3).

6.2.1 DECnet/OSI Networking Software

The OpenVMS operating system participates in the DECnet/OSI network through its networking interface: DECnet/OSI software. In addition, optional products that run on the OpenVMS system provide networking functions that augment the DECnet/OSI capability.

The two phases of DECnet available on OpenVMS VAX and OpenVMS Alpha systems are Phase IV and Phase V. DECnet for OpenVMS software, based on Phase IV DNA network architecture standards, supports the protocols for DECnet communications. DECnet for OpenVMS VAX software (previously known as DECnet–VAX Phase IV software) and DECnet for OpenVMS Alpha support similar features and are interoperable on a DECnet network. Phase IV implementations of DECnet can communicate with any other software that supports DECnet Phase IV protocols.

DECnet/OSI, which implements the latest phase of DNA (Phase V), provides the traditional DECnet features as well as conformance to international standards and recommendations approved by the following organizations:

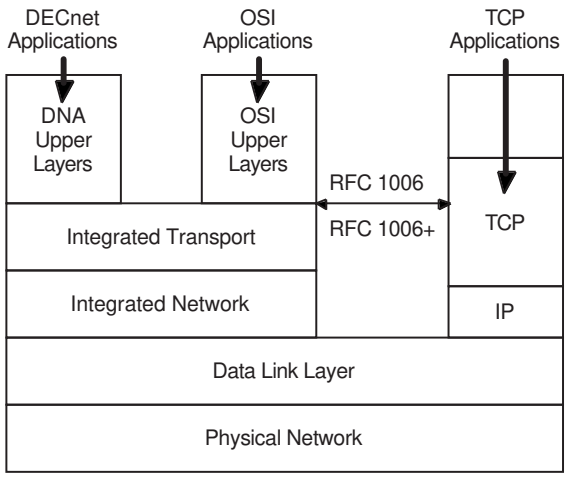
- International Organization for Standardization (ISO)
- International Telegraph and Telephone Consultative Committee (CCITT)
- Institute for Electrical and Electronics Engineers (IEEE)
- Internet Engineering Task Force (IETF)

DECnet/OSI supports the ISO standards defining the Open Systems Interconnection (OSI) model; these standards allow computer systems from different vendors to be interconnected. DECnet/OSI features a full OSI implementation with GOSIP certification. DECnet/OSI also supports the IETF Request for Comments (RFC 1006 and RFC 1006 Plus) that specify how OSI and DECnet transports should work over TCP transports.

As shown in Figure 6–1, DECnet/OSI integrates DECnet and OSI and supports links to TCP/IP protocols. The figure shows the DECnet/OSI layered architectural model that defines an environment for open networking. This model spans integration from the physical link to the application layer. The integration is provided in the lower layers where a variety of protocols can be shared by the upper layer applications. In the upper layers, separate protocol towers are maintained to provide specific services to network applications.

DECnet/OSI for OpenVMS supports the networking protocols that permit connection to other systems that use DECnet and to other vendor systems that support OSI. Existing DECnet Phase IV systems, applications, and network components can continue to function in a DECnet/OSI environment. DECnet/OSI end systems can communicate with, and remotely manage, DECnet Phase IV nodes in the same network. DECnet/OSI end systems can transmit and receive messages, but cannot route messages through the network. Multiprotocol routers perform message routing functions in DECnet/OSI networks (see Section 6.2.4).

Figure 6–1 Integration of DECnet, OSI, and TCP/IP Network Architectures



6.2.1.1 DECnet/OSI Features and Software

As implemented on the VAX and Alpha platforms, DECnet/OSI for OpenVMS provides significant features that expand DECnet networking capability:

- Incorporates OSI capabilities to support open, multivendor networks.
- Permits DECnet and OSI applications to run over TCP/IP protocols.
- Provides support for very large networks. A network based on DECnet/OSI is, in effect, not limited in size. DECnet/OSI introduces increased addressing capabilities for large networks.
- Supports the following options for storing node names:
  - Expanded Local Naming Services, a local option that permits up to 100,000 node names to be stored in a DECnet/OSI end system.
  - DECdns, a networkwide distributed name service that automates node name management, mapping node names to addresses. DECdns allows users to use network resources without knowing the network address of the resources. (Use of a networkwide namespace requires at least one DECdns server.)
  - DNS/BIND, the Domain Name System for TCP/IP, which provides for storage of IP (Internet Protocol) addresses and for use of node synonyms.
- Uses DECdts time service to provide a precise, fault-tolerant clock synchronization for systems in a local and wide area network.
- Uses a network management architecture that defines manageable units as entities. Network management is modular and distributed, permitting remote management of all network functions.

DECnet/OSI for OpenVMS includes the following software that conforms to ISO standards:

- OSI Transport Service software, which permits OSI applications to run on OpenVMS hosts

- OSI Applications Kernel (OSAK) software, which implements the upper three layers of the OSI Reference Model
- File Transfer, Access, and Management Service (FTAM), which provides the ability to access and transfer files in an open systems environment
- DECnet/OSI for OpenVMS Virtual Terminal (VT) software, which permits remote logins and access to remote applications on any system that runs an ISO-compliant VT implementation
- OSI application programming interfaces (APIs) to allow for development of OSI applications

Other optional software products related to DECnet/OSI that can run on OpenVMS include:

- Message Router and Message Router X.400 Gateway, which provides the ability to exchange mail in an open systems environment, and MAILbus 400 software, which supports X.400 messaging services
- DEComni, which provides an API for connection and management of manufacturing/utility control devices and systems, and DEComni MMS, which implements the ISO Manufacturing Message Specification (MMS)
- DEC/EDI, which enables users to electronically exchange structured business documents conforming to EDI (Electronic Data Interchange) standards using communications options including OSI

DECnet/OSI software, running on VAX and Alpha nodes, provides wide area network support, permitting connection to packet-switching data networks (PSDNs) that conform to the CCITT X.25 recommendations (such as TYMNET). For additional information, refer to the DECnet/OSI documentation and Software Product Descriptions.

OpenVMS users can use DECnet/OSI and PATHWORKS software to connect to PC LANs.

DECnet/OSI users can also access IBM SNA networks by means of Digital IBM Interconnect products, a family of networking products that include DECnet/SNA gateways and 3270 terminal emulators.

6.2.1.2 DECnet Network Management Tasks

General techniques for managing the network are summarized in Section 2.4.2. This section describes the tasks involved in managing DECnet Phase IV and DECnet/OSI nodes in a network. Phase IV and Phase V support different command languages to perform network management tasks.

Phase IV network management involves planning, building, tuning, and controlling DECnet networks through the use of NCP, the Phase IV command language. NCP functions include displaying statistical and error information, controlling network components, and testing network operation. NCP can be used to create and manage networks, including local and remote node operations, circuits, lines, and objects.

DECnet/OSI network management is based on the director-entity model described in Section 6.3.6. The entity-based network management architecture allows local and remote management of entities throughout the network. The command language for DECnet/OSI is the Network Control Language (NCL), which can be accessed either through a traditional command line interface or a graphical user interface (GUI). NCL accesses management directives defined for DECnet/OSI entities using the Common Management Information Protocol (CMIP), based on

ISO standards for encoding network management operations. NCL can perform basic management of all entities for a distributed system from a single location anywhere in the DECnet/OSI network. DECnet/OSI users can also use the DECnet Phase IV command line interface, NCP, to manage Phase IV nodes in the network. The NCP Emulator tool is used to install and use layered products that issue NCP commands.

Network management tasks for DECnet/OSI include the following:

- Management tasks:
  - Managing nodes in the local namespace and in the distributed namespace; managing the namespace
  - Setting up routing between DECnet Phase IV and DECnet/OSI areas
  - Downline loading, upline dumping, and controlling remote or unattended systems, using NCL and the Maintenance Operations Protocol (MOP)
  - Setting up network security, communications links, and the VMScluster alias, using NCL
- Monitoring tasks:
  - Reporting network events during network operation, using NCL and the Event Dispatcher
  - Collecting information about the local network configuration
  - Displaying network status and characteristics, using NCL
- Problem-solving tasks:
  - Testing network software and hardware, using NCL and Loopback tests
  - Testing network connections using DECnet Test Sender/DECnet Test Receiver (DTS/DTR)
  - Collecting and displaying information about specific protocol exchanges between systems in the network, using the Common Trace Facility (CTF)

DECnet/OSI network management support tools include a DECnet/OSI register tool used for naming and a DECnet/OSI migration tool to assist users in performing network management tasks and learning NCL.

OpenVMS users can perform the same general-user functions over the network that they can carry out for the local system, including remote file operations (see Section 5.5.3). They can enter DCL commands to perform operations over the network and develop command procedures and application programs to run over the network. In addition, they can send mail to remote nodes and use facilities like the Notes conferences to share data. Examples of DECnet/OSI applications software used for general-user operations on the OpenVMS system are listed in Table 6–1.

Table 6–1 Examples of DECnet/OSI Applications for General-User Operations

DECnet Application	Function
File Access Listener	Allows heterogeneous file transfer over the network using the Data Access Protocol (DAP)
DECdfs	Provides OpenVMS users with high-speed, transparent access to files stored on remote disks as if they were on local disks
FTAM	Provides the ability to access and transfer files in an open systems environment
Network Virtual Terminal	Supports the ability to set host to another system, using the Communications Terminal Protocol (CTERM)
MAIL-11	Provides mail service over the DECnet network
DEC Notes	Allows computer conferencing

6.2.2 OpenVMS Connections to TCP/IP Networks

The Digital TCP/IP Services for OpenVMS product (formerly called the VMS/ULTRIX Connection product) promotes interoperability and resource sharing between OpenVMS VAX and OpenVMS Alpha systems and other systems running TCP/IP. This product makes OpenVMS a full participant in a TCP/IP network. It supports networking, file sharing, remote terminal access, electronic mail, and application development between OpenVMS systems and UNIX systems. Major components of the Digital TCP/IP Services for OpenVMS are listed in Table 6–2.

Table 6–2 Components of Digital TCP/IP Services for OpenVMS

Component	Description
Set of ARPANET[1] communication system protocols	Supports industry-standard networking on OpenVMS through protocols based on the 4.3 Berkeley Software Distribution: TCP (Transmission Control Protocol) IP (Internet Protocol) FTP (File Transfer Protocol) Telnet Protocol UDP (User Datagram Protocol) Other protocols (ARP, ICMP, RIP)
Remote Procedure Call	Allows application developers to partition applications along subroutine interfaces and have those subroutines execute on remote hosts
NFS client software	Allows OpenVMS users to mount NFS files on other hosts as if they were local OpenVMS files
NFS server software	Permits UNIX clients to access OpenVMS files and files compatible with UNIX that are stored on OpenVMS
Anonymous FTP	Allows remote users without an account on a specific node to copy files to and from that node

[1]The network on which TCP/IP was first implemented.

(continued on next page)

Table 6–2 (Cont.) Components of Digital TCP/IP Services for OpenVMS

Component	Description
System management and DECwindows interfaces	Permits the OpenVMS system manager to manage Internet communications and the NFS server without detailed knowledge of UNIX networking

Users can connect OpenVMS and Digital UNIX systems with UNIX systems from other vendors, using industry-standard protocols for communication. Users can also write network applications that access the Internet protocols using standard OpenVMS services, including a QIO programming interface to access the lower level protocols.

In combination with DECnet/OSI, TCP/IP products from Digital and other vendors support running DECnet and OSI applications over TCP/IP transport connections.

OpenVMS systems running TCP/IP layered products can communicate over the global Internet. For example, in an open, multiprotocol network, OpenVMS systems on LANs can use a corporate gateway that runs TCP/IP or a wide area multiprotocol router to connect to the Internet.

Network File System (NFS) for OpenVMS provides NFS client and server software on OpenVMS systems. NFS is a protocol that gives network clients access to remote file services. The NFS client software allows OpenVMS users to access remote NFS files. The NFS server software enhances data sharing among UNIX clients by providing a central data storage facility for OpenVMS and UNIX files. A UNIX client can access OpenVMS files or files compatible with UNIX that are stored on the OpenVMS server (which can be either an OpenVMS system or a VMScluster system).

6.2.3 Network Security

OpenVMS VAX and OpenVMS Alpha security features protect DECnet/OSI nodes, user accounts, and data through password protection and file protection mechanisms. In general, the system manager can control access to a system through the use of proxy accounts and default accounts. The system manager can use the AUTHORIZE command to create proxy accounts, which specify remote users or groups of users who can access data on the local system with the same privileges as users logged in to the local system.

For DECnet for OpenVMS (Phase IV), the establishment of logical links with remote nodes can be controlled through specification in the network database of the allowed logical link connections.

To help protect the security of messages transmitted over a local area network, the Ethernet Enhanced Security System can selectively encrypt information across a network, without affecting other network nodes not requiring encryption. This system includes the Digital Ethernet Secure Network Controller (DESNC) and the Key Distribution Center (KDC) system attached to the LAN. The KDC system provides tools to permit a system security manager to implement and manage a security-enhanced LAN. The DESNC and KDC system implements a default access control policy that allows all nodes on a LAN network to communicate with each other after successful authentication.

6.2.4 Other Supported Networking Protocols and Products

On local area networks, OpenVMS supports the following protocols:

- The LAT protocol, a highly efficient local area transport service that allows for connections to Digital and nonhost systems. The LAT protocol is used to communicate with other nodes and with devices (terminals, modems, or printers) offered by terminal servers on a local area network.
- The local area disk (LAD) protocol, used to access compact disc media that reside on a Digital InfoServer system.
- The local area systems transport (LAST) protocol, used for virtual disk access.

OpenVMS systems can use the LAD/LAST protocol to access a library of OpenVMS software and online documentation stored on CD–ROMs on the InfoServer. The CD–ROM (compact disc read-only memory) is an optical data storage technology formatted according to the ISO 9660 standard. The InfoServer is a dedicated hardware and software device that supports locally attached CD–ROMs and provides shared read-only access to CD–ROM discs. The InfoServer also provides support for Initial System Loading (ISL) via the LAN, permitting the OpenVMS operating system to be loaded from a compact disc.

OpenVMS hosts and other hosts can be connected to terminals over the LAN by means of terminal servers (the DECserver series). For example, a multiprotocol terminal server offers high-speed connections for terminals, printers, PCs, and other devices to any network service that supports LAT or TCP/IP. This DECserver can be used in OpenVMS, Digital UNIX, and DOS environments or in a network that combines these operating systems. This terminal server can be directly managed in multivendor network environments as a standalone unit or in a DEChub configuration. The server provides multivendor connectivity and wide area routable terminal service.

Multiprotocol wide area networking routers support routing in multivendor networks. The DEC WANrouter and the DECNIS network integration server are multiprotocol routers that provide a single routing service for DECnet/OSI nodes, DECnet Phase IV nodes, TCP/IP nodes, and OSI end systems. Some of these routers also implement X.25 routing circuits. The WANrouter and DECNIS use the Integrated IS-IS (Intermediate System-to-Intermediate System) routing protocol that conforms to the OSI model.

Digital also provides an open, intelligent crossbar switch, the GIGAswitch device. The GIGAswitch is a datalink-independent device that supports multiple, dynamic, simultaneous FDDI connections for Alpha, VAX, and other vendors' processors. The GIGAswitch facilitates the move to a high-bandwidth switched network that meets the demands for such applications as imaging and multimedia.

6.3 Multivendor Integration Using Middleware

Middleware products support multivendor computing capabilities in a distributed networking environment. Middleware products run on OpenVMS and other Digital systems and on systems from multiple vendors. Middleware products, built using industry standards, promote multivendor interoperability (as discussed in Section 2.1.5.2) and provide services for distributed applications (as described in Section 2.2.3).

Middleware products and tools help applications from different vendors work together. Middleware facilitates communication between applications so that different units of an enterprise or business can share information quickly and easily, even though the units may be distributed in different locations and may use different computer systems. The services supplied by middleware help make it possible for a distributed multivendor environment to appear to the user as an integrated environment.

Middleware services are available for OpenVMS systems, other Digital systems, and many other systems from multiple vendors. Digital offers middleware services and frameworks on a number of platforms, including the following:

- OpenVMS VAX systems
- OpenVMS Alpha systems
- Digital UNIX Alpha systems
- ULTRIX RISC systems
- MS–DOS systems
- Microsoft Windows systems
- Windows NT systems
- Macintosh systems

In addition, middleware services are available from Digital on other platforms, including:

- SunOS systems
- OS/2 systems
- HP–UX systems
- AIX for the RS/6000 systems
- SCO UNIX systems
- ULTRIX MIPS systems

Refer to *The Middleware Source Book* for a full description of middleware.

6.3.1 Using Middleware with Applications

Application developers can use middleware services to create applications that provide transparent access to resources throughout a heterogeneous network.

To facilitate integration among different systems, middleware uses a client/server model of computing combined with standards-based application programming interfaces (APIs) and networking protocols. For example, an application (a client) requests information from another application (a server). The request is made through the API. The API communicates with the underlying middleware service, which executes the client's request. The process occurs transparently even though the client and server are on different systems on the network. The APIs and network protocols, which pass data and control between client and server, implement industry and international standards, making the applications less dependent on any specific platform.

6.3.2 Middleware Support for Industry Standards

Middleware services and their interfaces are based on open, industry standards whenever possible, including standards created by the International Organization for Standardization (ISO), the American National Standards Institute (ANSI), and the Institute of Electrical and Electronics Engineers (IEEE). IEEE standards include the POSIX 1003.n standards. The use of open standards enables middleware services to span the range of platforms and networks commonly found in enterprisewide information systems.

The middleware architecture is influenced by a number of specifications, including the Open Software Foundation (OSF) Application Environment Specification (AES) and the X/Open Common Application Environment (CAE) specification.

Other consortia and groups influencing one or more of the middleware services include the Internet Engineering Task Force (IETF), the Object Management Group (OMG), and the MIT X Consortium.

6.3.3 Middleware Service Categories

Middleware consists of a large number of services, grouped into the following six categories:

- Presentation services

  These services let an application interact with users. Some services operate across a variety of interface devices, while others support only a particular class of interface device.

- Communication services

  These services let an application communicate with other applications, both local and remote. They include services for electronic mail and various communication protocols.

- Control services

  These services permit an application to control program execution in both local and distributed system environments. They include services to invoke functions provided by application programs, making for a highly extensible system.

- Information services

  These services enable the application to define, store, access, and manipulate data. They provide a shared repository of information for tools and applications and for individuals and organizations in an enterprise. These services support multiple access mechanisms for distributed information.

- Computation services

  These services let an application perform complex operations on data. These services support international character handling and provide a networkwide concept of time for application programs, which lets applications manipulate time values and synchronize activities consistently.

- Management services

  These services enable system managers to manage an application easily and consistently from remote nodes. Middleware supports the same scheme to manage applications as it uses for all of the middleware system components. This enables system managers to manage any component from anywhere in the network in a consistent fashion.

Table 6–3 summarizes the middleware services and frameworks and their Digital product implementations. Digital products listed also run individually on OpenVMS operating systems unless otherwise indicated.

Table 6–3 Middleware Services and Frameworks and Their Digital Implementations

Middleware Services or Frameworks	Digital Implementations
Presentation Services	
Forms service	DECforms
Graphics service (GKS)	Digital GKS
Graphics service (PHIGS)	Digital PHIGS
Motif service	DECwindows Motif
Printing services	BSD Print System DQS PATHWORKS printing support PrintServer
Terminal services	DECwindows, PATHWORKS, Telnet, and other terminal emulators
X Window System service	DECwindows Motif
Communication Services	
Messaging services	MAILbus 400 MTA MailWorks for OpenVMS Server
Message queuing service	DECmessageQ
Remote procedure call service	DCE RPC
Control Services	
Continuous computing services	Reliable Transaction Router
Object broker services	ObjectBroker (formerly called DEC ACA Services)
Multithreading service	DECthreads
Transaction management service	DECdtm

(continued on next page)

Table 6–3 (Cont.) Middleware Services and Frameworks and Their Digital Implementations

Middleware Services or Frameworks	Digital Implementations
Information Services	
Compound document service	CDA Run-Time Services
Data access services	DEC DB Integrator DEC DB Integrator Gateway
Directory services	DNS (Domain Name System), DNS/BIND DCE CDS (Cell Directory Service) DECdns X.500 directory service
File-sharing services	NFS network file system DECdfs PATHWORKS disk/file sharing service
File Transfer, Access, and Management Service	DECnet/OSI FTAM
Computation Services	
Distributed time service	DECdts DCE DTS
Internationalization services	XPG3, XPG4 internationalization services
Management Services	
Management Agent service	Common Agent (runs on Digital UNIX)
Security service	DCE Security
Personal Computing Integration Service	
PATHWORKS services	PATHWORKS products
Frameworks	
Electronic Data Interchange framework	DEC/EDI FileBridge for DEC/EDI HostBridge for DEC/EDI
Management director framework	POLYCENTER Director Framework
Transaction-processing monitor framework	Desktop for ACMS
Workgroup framework	LinkWorks

6.3.4 Distributed Computing Environment Software

The OSF Distributed Computing Environment (DCE), described in Section 2.2.4, is a standard set of software services and interfaces that support the creation, use, and maintenance of distributed client/server applications. Digital has implemented a family of DCE products that include a certified set of DCE functions along with software for developing distributed applications.

Digital DCE supplies system managers with a set of tools to consistently manage the entire distributed computing environment while also ensuring the integrity of the enterprise. The Digital DCE product family provides users with maximum flexibility for configuring the environment, which is known as a DCE cell.

Table 6–4 describes the functions supplied by the Digital DCE product family. The DCE for OpenVMS product family includes the following run-time services and application development software, which run on the OpenVMS VAX and OpenVMS Alpha operating systems:

- DCE Run-Time Services for OpenVMS, required for all systems participating in the DCE cells. The run-time services kit includes DCE client functions and DCE administration tools.
- DCE Application Development Kit for OpenVMS, required for those developing distributed applications but optional for other users. The kit provides users with an Interface Definition Language (IDL) for writing remote procedure calls.

The DCE product family supports two DCE servers. These servers are implemented for DCE on OpenVMS systems:

- DCE Cell Directory Server (CDS), a central repository containing information about the location of resources in the DCE cell. The CDS server allows access to resources by a single name, regardless of physical location. One CDS server is required for each DCE cell.
- DCE Security Server, which protects resources from unwanted access by providing authentication and authorization services and provides for secure communications within and between DCE cells. One Security Server is required for each DCE cell.

The DCE for OpenVMS family also supports the Resource Broker. The Resource Broker is a graphical tool that helps eliminate bottlenecks by letting an administrator examine server usage around the network and allocate servers for tasks based upon their availability and efficiency.

Table 6–4 Digital DCE Product Family Functionality

DCE Function	Description
DCE Remote Procedure Call (RPC)	Creates and runs client/server applications
DCE Distributed Time Service (DTS)	Provides synchronizaton of time in a distributed network environment
DCE Threads Service	Provides user context multiprocessing functionality
DCE Cell Directory Service	Provides location-independent naming for resources
DCE Security Service	Provides secure communications and access by means of authorization and authentication services
IDL Compiler and associated development components (such as UUIDGEN)	Supports IDL, an ANSI C-based language for writing remote procedure calls

(continued on next page)

Table 6–4 (Cont.) Digital DCE Product Family Functionality

DCE Function	Description
DCE Resource Broker	Supplies the location of servers on which services can run

The Digital DCE product family is an implementation of OSF DCE standards, adapted and enhanced for OpenVMS as follows:

- Simplified installation and configuration
- IDL support for both C and Fortran languages
- IDL development templates
- A conversion utility for DCE RPC programs
- The PC NSI Proxy Agent, which enables interoperability with Microsoft's PC RPC

The DCE for OpenVMS product family supports the following networking transports: TCP/IP, UDP, and DECnet Phase IV transports.

6.3.5 Application Development in Multivendor Environments

Digital's software development environment provides a set of software development solutions, based on standards, that enable software development, reengineering, deployment, and management on a variety of computing platforms. The software development environment is designed to bring the open systems benefits of flexibility, investment enhancement, and multivendor interoperability and portability to software development.

The software development environment permits developers to build applications in the OpenVMS, Digital UNIX, or PC environment for deployment across a network of systems from different vendors.

A variety of tools, developed by Digital and by other vendors, can be used to support the entire software development life cycle. Tools from multiple sources are integrated into a single development environment through a framework of the following levels of integration:

- Presentation integration: DECwindows Motif provides a common look and feel across tools. It allows programmers to interact with different tools in the same way.
- Control integration: ObjectBroker (formerly called DEC ACA Services) manages the flow from one activity to another, providing dynamic, functional integration. ObjectBroker integrates functions from one tool to the next, permitting users to move from one activity to another in a natural manner.

  ObjectBroker is Digital's Object Request Broker, complying with the Common Object Request Broker Architecture (CORBA) specification, the standard for central communication and integration of software objects developed by the Object Management Group (OMG). Object-oriented programming permits objects (modular pieces of software) to be joined together to create programs. ObjectBroker allows developers to use an object-oriented paradigm to integrate independently developed applications across heterogeneous environments. ObjectBroker facilitates the transition to client/server computing and reduces the cost of developing new client/server applications.

The software development environment provides several DECset software tools that support software coding, testing, and maintenance of applications and data on OpenVMS VAX and OpenVMS Alpha systems and Digital UNIX systems. The DECset tools include:

- DEC Language-Sensitive Editor/Source Code Analyzer (LSE/SCA), used to develop and debug source code modules for applications. The LSE/SCA editor has knowledge of the syntax of Digital programming languages and can provide context-sensitive help. LSE/SCA also allows interactive inquiries about the structure of the program.
- DEC Code Management System (CMS), which tracks everything that happens to project files during development, including revisions of source code, documentation, data files, test files, system build descriptions, and requirements documents.
- DEC Test Manager, which simplifies the testing process by automating the organization and execution of tests.
- DEC Module Management System (MMS), which automatically rebuilds the system after programmers make changes to application code.
- DEC Performance and Coverage Analyzer (PCA), which collects and analyzes performance and test information; used to identify possible performance problems in an application.

Other development tools used in the software development environment include:

- DECADMIRE, which provides a means of generating applications that are based on ACMS or DECforms and database products supplied by other vendors. DECADMIRE supports rapid development of client/server applications and makes complex transaction-processing applications easy to develop.
- Forté, an integrated set of tools for developing, deploying, and managing client/server distributed applications. The product generates distributed applications, including native client and server program components. Applications developed with Forté can run on multiple platforms and can provide access to databases such as Oracle and Sybase databases.
- DECplan, which is an integrated time and project management tool designed to help users plan, track, schedule, and report on projects and meetings. DECplan is a client/server tool based on DECwindows, useful in large, heterogeneous networks.

Digital's software development environment supports key programming languages and compilers, as described in Section 4.2.2.

In addition, the Digital software development environment supports industry-standard tools for developing three-dimensional (3D) graphics applications. The following tools run on both OpenVMS VAX and OpenVMS Alpha systems:

- Digital GKS graphics interface services that implement the ANSI and ISO standards for GKS–3D (the Graphical Kernel System for Three Dimensions). Digital GKS lets developers create device-independent two- and three-dimensional image-manipulation applications.

- Digital PHIGS graphics interface services that implement the ANSI/ISO standard for PHIGS (the Programmer's Hierarchical Interactive Graphics System). Digital PHIGS lets developers create device-independent three-dimensional model-manipulation applications.

An additional three-dimensional software product, Digital Open3D for OpenVMS Alpha, supports the PXG family of graphic accelerators on Digital's Alpha workstations. This software includes programming libraries for developing new applications. Digital Open3D with DECwindows Motif provides an environment for developing applications that include two- and three-dimensional graphics.

6.3.6 Managing Enterprisewide Multivendor Environments

The comprehensive approach to network management used by Digital networks also extends to other vendors, networks, systems, and technologies. Digital products provide for effective management of evolving, heterogeneous, multiprotocol, multivendor enterprises.

Network management for an enterprise is based on models for integrated management systems and self-managing network and system components and lays out the director/entity framework for implementing automated management functions. The two major components are the manageable units called entities and the directors that manage the entities.

The POLYCENTER solution for enterprise management, mentioned in Section 2.4.3, is built on middleware services. The basic POLYCENTER option includes software packages that provide base-level management capabilities in scheduling, fault and problem management, security, and storage management. Advanced network management applications perform higher level tasks such as integrating network information, loading it into databases, and presenting it to managers in graphic form.

The POLYCENTER framework is an extendable platform on which other POLYCENTER products, third-party products, or user-development management modules can be added. This multiplatform software system can be extended to manage practically any entity across a distributed environment.

Other POLYCENTER products are described in Chapter 7.

6.4 OpenVMS Software in Multivendor Client/Server Environments

Client/server configurations provide a means of making a distributed, multivendor environment more effective. The client/server style of distributed computing (as described in Section 2.2.1) is highly flexible and adapable to a wide variety of software and hardware configurations. Servers make it possible to share resources, use data and applications on different vendors' systems, and provide the power of large systems to small client computers. Client systems can be integrated into enterprisewide networking systems, obtaining access to the resources of an enterprise while still retaining their own individual computing environments.

The following sections describe how OpenVMS client/server software is used in VMSclusters and how an OpenVMS system functions as a server to PCs in a PATHWORKS multivendor environment.

6.4.1 VMScluster Servers and OpenVMS Clients

In any VMScluster system, users can share computing, disk and tape storage, and batch and print job processing resources. Any node in the cluster can use clusterwide batch and print queues. VMScluster technology also allows cooperating OpenVMS systems to share file and print resources over a LAN. This capability provides a mechanism for offering print and computing services to the network.

In a VMScluster configuration, the mass storage control protocol server and the tape mass storage control protocol server make locally connected disks and tapes available across the cluster. A VMScluster system can serve files to the LAN for use by cluster members.

Another client/server relationship on a VMScluster involves DECwindows Motif display services. These display services offer a consistent interface for sharing display resources across a VMScluster system. Users can display the output of programs running on any system in the cluster or any other system in the network.

VMScluster systems can also act as servers to clients on other vendors' systems in PATHWORKS configurations (as described in Section 6.4.2).

6.4.2 OpenVMS Servers in PATHWORKS Environments

PATHWORKS software permits the creation of an integrated desktop environment in which PC users can share information and resources on a LAN and access applications, data, and resources on OpenVMS servers and other servers across local and wide area networks.

The PATHWORKS family of network operating system software includes server software and client software: PATHWORKS server software running on OpenVMS VAX, OpenVMS Alpha, and UNIX based operating systems, and PATHWORKS client software running on personal computers and Macintosh systems.

To function as a server to PC clients, the OpenVMS VAX and OpenVMS Alpha systems use PATHWORKS for OpenVMS software in conjunction with the appropriate networking software products (for example, DECnet/OSI software and TCP/IP products such as Digital TCP/IP Services for OpenVMS).

Clients located on LANs use various networking transport protocols to communicate with servers. PATHWORKS network operating system (NOS) technologies provide basic network services.

PATHWORKS offers a choice of servers, clients, LAN technologies, network transports, and network operating systems, as follows:

- Servers include OpenVMS operating systems (running on all of the VAX and Alpha processors) and ULTRIX, Digital UNIX, and SCO UNIX operating systems.
- Clients include DOS, Windows, Windows 95, Windows NT, OS/2, and Macintosh systems.
- LAN physical media types include Ethernet, FDDI, and token ring.
- Network transports include DECnet, TCP/IP, IPX, NetBEUI, and AppleTalk, as well as access to X.25 network connections.

- Network operating system technologies include basic network services for LAN Manager, NetWare, and AppleShare.

PATHWORKS client/server software is described in Section 6.4.3. Networking connections between PATHWORKS clients and servers are covered in Section 6.4.4. Services provided by OpenVMS servers to PC clients are discussed in Section 6.4.5.

6.4.3 PATHWORKS Server and Client Software

OpenVMS servers running PATHWORKS for OpenVMS software can act as file, print, and mail servers to PC clients.

PATHWORKS for OpenVMS servers include:

- PATHWORKS V5 for OpenVMS (LAN Manager): PATHWORKS can use OpenVMS technology called clustering. Clustering allows multiple systems to behave as a single server and offers greater resiliency, more efficient load balancing, and easier system management of the server.

  OpenVMS servers support Microsoft LAN Manager V2.2 NOS file and print services to DOS, Windows, Windows 95, Windows for Workgroups, Windows NT, or OS/2 clients using DECnet, TCP/IP, and NetBEUI connections.

- PATHWORKS for OpenVMS (NetWare): OpenVMS servers can be configured to provide services in NetWare networks. One server can be connected to up to eight NetWare networks. NetWare networks support the IPX protocol.

- PATHWORKS for OpenVMS (Macintosh): OpenVMS servers can act as file, print, and mail servers to Macintosh clients. Client and server share information and resources using AppleTalk and DECnet or TCP/IP connections.

- PATHWORKS for Windows NT: Microsoft RAS software has been enhanced to allow remote clients to connect to DECnet file servers.

- PATHWORKS for Digital UNIX (LAN Manager): Digital UNIX servers can act as file, print, and mail servers to PC clients over DECnet or TCP/IP connections.

- PATHWORKS for Digital UNIX (NetWare): Digital UNIX servers can be configured to provide NetWare services. Clients can connect with NetBEUI.

- PACER for Digital UNIX: Digital UNIX servers can act as file, print, and mail servers to Macintosh clients. Client and server share information and resources using AppleTalk and DECnet or TCP/IP connections.

PATHWORKS client software allows PCs and Macintosh systems to connect to Digital's PATHWORKS servers and industry standard NOS servers—such as those from Novell (NetWare) or Microsoft (Windows NT)—for file and print services.

In addition, PATHWORKS client software provides the following:

- For Windows and DOS:
  - Support for Windows 95
  - Attribute-based searches for printers—for example, by color or location, or both
  - A simple method for organizing and sharing information residing in diverse locations for ad hoc workgroups

  - "Protected mode" DECnet and its utilities, saving conventional memory
  - TCP/IP and its utilities
  - Middleware applications such as DECmessageQ and ObjectBroker
  - Mosaic, an Internet browser
  - PATHWORKS Mail, a front end to OpenVMS Mail
  - VT320 terminal emulator
  - ManageWORKS and management applications
  - And more
- For Windows NT:
  - DECnet and its utilities
  - MAIL-11 Client
- For the Macintosh:
  - DECnet and its utilities
  - TCP/IP utilities (FTP and others)
  - Terminal emulators
  - X display server
  - Remote management for PATHWORKS for OpenVMS (Macintosh) server
- For OS/2:
  - SETHOST, a terminal emulator
  - DECnet, LAT, and utilities
- For Windows 95—as part of PATHWORKS for DOS and Windows:
  - Mosaic, an Internet browser
  - "Protected mode" DECnet, LAT, and utilities, saving conventional memory
  - VT320 terminal emulator

PATHWORKS configurations are flexible and can be changed easily. In most cases, PATHWORKS client software is originally stored on the server and is downline loaded over the network to the PC. PC clients can be connected to multiple servers simultaneously.

Configuration changes, such as the addition of new PATHWORKS clients and servers, do not cause disruption to the PC user environment.

6.4.4 PATHWORKS Network Connectivity

PATHWORKS provides PC users with a family of network integration products that supply multivendor LAN connectivity, permitting users on one LAN to communicate with users on another LAN. PC users can also connect from LANs to the rest of an enterprise through WAN network connections.

In PATHWORKS configurations, PC clients on LAN physical media use networking transport protocol software to communicate with OpenVMS and other servers. Network operating systems (NOSs) provide basic networking services required for PATHWORKS connections.

Each server and client node that runs PATHWORKS software uses a network transport. PATHWORKS supports a broad range of network transports, as listed in Table 6–5, as well as the use of more than one transport at the same time.

Table 6–5 Network Transports Supported by PATHWORKS Software

Transport	Description
DECnet	Used to provide file and printer services to clients that run DECnet and connect to an OpenVMS server. DECnet supports connections to X.25 networks.
TCP/IP	Used to provide file and printer services to PATHWORKS clients that run TCP/IP and connect to an OpenVMS server as well as to many industry standard clients. TCP/IP also supports connections to Internet networks.
AppleTalk for OpenVMS	Implements AppleTalk protocols in the OpenVMS environment; used by Macintosh clients for file and print sharing.
IPX	The Novell NetWare Internetwork Packet eXchange protocol; IPX protocols can be encapsulated in DECnet packets for transmission across wide area DECnet connections.
NetBEUI	The transport layer of the network basic I/O system (NetBIOS) protocol that lets PC users connect to a PATHWORKS server as well as to an industry standard server such as the IBS LANserver.
LAT	The Digital local area transport used for terminal communications and print services for PC clients.

Other PATHWORKS for OpenVMS (Macintosh) connections include the AppleTalk to DECnet Gateway (OpenVMS software), which lets a Macintosh application connect to a DECnet service on any server.

The following types of physical interconnect technology can be used in the LANs to which the PCs are connected:

- Ethernet or FDDI, which provides connection between DOS, PC clients, and OpenVMS servers
- Token ring, which provides network connection between PC clients and OpenVMS servers equipped with TRNcontroller adapters
- LocalTalk, which provides connection of Macintosh clients to OpenVMS servers through gateways

Network operating system technologies include the file and print services for LAN Manager, NetWare, and AppleShare. Connectivity for LAN-based PCs and the OpenVMS and Digital UNIX servers are based on adaptations of LAN Manager from Microsoft. PATHWORKS client support for NetWare allows PCs to function simultaneously as PATHWORKS clients and NetWare workstations. The NOS for Macintosh systems is AppleShare (supplied by Apple Computer, Inc.); AppleShare requires that the AppleTalk network transport protocol be used as the file and print transport.

6.4.5 OpenVMS Services for PATHWORKS Clients

Individual OpenVMS systems and VMScluster systems can provide services to PCs. An OpenVMS system running the appropriate PATHWORKS software can act as a server for personal computers or as a server for Macintosh systems. More than one PATHWORKS server product can exist on an OpenVMS system at the same time.

In a PATHWORKS client/server configuration, OpenVMS servers can be required to handle multiple requests coming from many systems at different times, and must handle requests quickly to ensure good response time at the personal computer. In turn, the desktop systems must be served with the proper security and system management protection. Servers may offer data of a sensitive nature (for example, an OpenVMS server may provide financial spreadsheet data to an MS–DOS PC).

The following characteristics of the OpenVMS system make it an effective server for desktop PCs:

- OpenVMS is a multiuser system and resource server.
- OpenVMS features VMSclusters, which ensure system reliability and scalability
- OpenVMS is a multiprogramming, multiprocessing system with preemptive priority scheduling and is capable of responding rapidly to many requests from different clients.
- OpenVMS is enhanced with security and system management tools and techniques based on years of supporting timesharing users.
- OpenVMS resource serving over the network is a natural evolution from resource serving to multiple users logged in to the system in a timesharing environment.
- OpenVMS software and layered products provide extensive support for managing a broad range of systems and complicated networks.

Using PATHWORKS to link PC clients with OpenVMS servers over a LAN permits the PC users to share files, applications, printers, and electronic mail. PC users can also take advantage of wide area networking capabilities, system management, and security features provided by the OpenVMS server, as well as administrative and security controls provided by LAN Manager.

Specifically, OpenVMS servers running PATHWORKS software can provide the following services to PC and Macintosh clients running the appropriate PATHWORKS software:

- File and print services
- Electronic mail services
- VMScluster access
- Management, security, and resource control

Other services provided to PC clients by OpenVMS servers are described in Section 6.4.6.

With the appropriate PATHWORKS software, PC and Macintosh clients can access and store files on OpenVMS and VMScluster systems anywhere in the local or wide area network. The remote file service appears to the PC client as a transparent extension of the PC's local disk. The PC files are automatically stored on an OpenVMS server in OpenVMS file format and can be shared transparently with users on DOS, Windows, Windows 95, Windows NT, OS/2, and Macintosh systems. Applications stored on OpenVMS servers can be run on the client, using the resources of the client. OpenVMS servers provide large system file capacity, data integrity, and security to PC files.

OpenVMS servers provide print services to PC clients using standard OpenVMS print queues. PC users can print files on a local printer connected to the PC or a remote printer connected to the OpenVMS server on a PATHWORKS network. PATHWORKS for OpenVMS (Macintosh) provides DECshare print services to Macintosh clients.

6.4.5.1 Mail Services for PC Users

PATHWORKS mail services provide full OpenVMS mail-handling capabilities to PC users. Using MAIL, the PC user can communicate with other PC users on the LAN and with PC and non-PC users throughout the enterprise. OpenVMS mail services support distribution lists, real-time mail notification, and other mail functions.

PC users can also access MailWorks for OpenVMS (formerly known as ALL–IN–1 MAIL) on OpenVMS servers. MailWorks for OpenVMS conforms to the ISO /CCITT X.400 user agent standard and permits the user to exchange mail with users of any public or private mail system that complies with the X.400 standard for message exchange throughout the world.

Users of basic Mail and MailWorks for OpenVMS can exchange messages with X.400 messaging systems such as MCI Mail, facsimile systems, and proprietary mail systems such as IBM PROFS, as well as with Digital UNIX systems.

6.4.5.2 VMScluster Access for PC Clients

The PATHWORKS V5 for OpenVMS server provides for operations in VMScluster and VAXcluster environments. The PATHWORKS server includes support for the PATHWORKS cluster alias name service; this transport-independent service allows multiple cluster members to appear as a single server to the connecting clients. The PATHWORKS cluster alias name is further enhanced with a load-balancing feature that connects clients to the least loaded cluster member in a VMScluster at connection time.

6.4.5.3 OpenVMS Management Services for PC Clients

Additional services are available to PC users through PATHWORKS software:

- Security for PC data: The OpenVMS server provides password protection, access control, standard file protection, and security. Use of passwords permits control of access to network file and print resources. Passwords can be associated with file directories or printers being offered for use by OpenVMS servers. Security at the level provided by OpenVMS systems can be applied to PC files, including access control lists that define users' rights to access resources or services. Service and file access controls can be specified across a VMScluster system, as well as on individual OpenVMS servers.

PATHWORKS V5 for OpenVMS supports a security model that accommodates LAN Manager security as well as OpenVMS host security. LAN Manager servers support two security levels: user-level controls access based on a user's account and password, and share-level controls access based on the password associated with a resource.

- Backup of PC hard disks: The OpenVMS server provides for unattended backup of PC hard disks through the use of timed batch files. During backup, PC file-access utilities prohibit unauthorized access.
- Management: The NetWare command line software in PATHWORKS V5 for OpenVMS provides PC users with command line services for managing and controlling the PATHWORKS network, including control of client connection to the server.

  PATHWORKS V5 LAN Manager provides domain services to control user access to the network and manage large diverse networks. Domain services also support the use of a single login request to gain access to many servers on a LAN.

  ManageWORKS is a Windows PC-based solution which can manage Digital, IBM, Microsoft, and Novell network operating systems and network hardware devices on LANs.
- Administrator's utilities: These utilities permit the system administrator to register clients, specify clients for remote booting, record network and server events, broadcast messages to PC users, and adjust server parameters to achieve efficient and reliable performance.

  LAN Manager servers can be managed using a Windows based network management application included with the PATHWORKS for DOS and Windows client.
- Remote boot services for DOS PCs: The remote boot capability permits DOS PCs to activate their operating system, startup files, and PC networking software from the OpenVMS server on which the software is stored. Remote booting allows control of PC resources shared by networked PCs and facilitates system administration of multiple PCs.

6.4.6 Other OpenVMS Services for PC Users

OpenVMS servers running additional optional software can act as database and transaction-processing servers to personal computer clients.

PC users can use data access services to gain transparent access to remote databases on OpenVMS servers. Database services based on SQL requests allow PC users to retrieve corporate data and use the input in PC applications. Examples of large databases on OpenVMS systems that PCs can access are the RMS database and other vendor relational databases (such as Oracle Rdb).

The Digital ACCESSWORKS products can be used to provide preconfigured client/server solutions based on the commercial strength of OpenVMS systems. DEC ACCESSWORKS supports a range of client devices, including OpenVMS, UNIX, MS–DOS, Windows, SPARCstation, OS/2, and Macintosh devices. It offers users concurrent, transparent access to a variety of multivendor data sources, such as OpenVMS RMS, Oracle Rdb, and IBM DB2, VSAM, and IMS databases. The various ACCESSWORKS solutions provide core software tools, middleware, and server software, including PATHWORKS for OpenVMS network operating system software and DECnet and TCP/IP software.

PC users can act as full participating clients in transaction-processing applications through the DECtp Desktop for ACMS product. The ACMS transaction-processing monitor running on OpenVMS is used to define, run, and control transaction-processing applications. Desktop ACMS software enables PC, Macintosh, and OpenVMS clients to interact with ACMS applications on the OpenVMS server that accesses the database. The clients establish networking connections to system servers using PATHWORKS software and DECnet, TCP/IP, or NetWare transports.

Desktop ACMS software links back-end application functionality to front-end forms management attributes such as displays, terminal characteristics, graphics, and color. ACMS applications can interact with many types of desktop computers and desktop user interface packages for Macintosh, MS–DOS, and Windows systems. Heterogeneous support enables installations to mix PC, Macintosh, and OpenVMS computers as front-end systems to the same ACMS application.

7

OpenVMS Systems in Commercial Environments

OpenVMS systems incorporate commercial-strength software capabilities that make them highly suitable for use in critical production system computing environments and in other configurations that require dependable, high-integrity software.

This chapter covers the capabilities of OpenVMS systems as distributed production servers in open environments. It also describes layered software supplied by Digital and other vendors that can run on OpenVMS systems and VMSclusters in distributed multivendor production environments.

7.1 OpenVMS Production Systems

OpenVMS systems provide high-integrity, commercial-strength capabilities that meet the needs of production systems. A production system is a computing system configuration essential or critical to the operation of an organization. Production systems can exist in a variety of computing environments, ranging from a single system, cluster, or client/server configuration to complex factory or commercial configurations involving desktop devices and mainframes.

A production system need not be limited to one system or cluster. A distributed production system can involve a collection of databases and processors, linked by high-speed, high-throughput networks, with enabling software that makes the configuration a single virtual database and processor to the end user, operator, and system manager.

OpenVMS systems incorporate powerful, dependable production system capabilities (as summarized in Section 2.3). In addition, OpenVMS systems support reliable distributed processing in integrated multivendor computing environments (as described in Chapter 6). OpenVMS systems can be used with optional software that runs on OpenVMS to meet the following needs of distributed production systems:

- The production system must remain up and running with little or no interruption.
- The system must be able to produce data, gathering information from the real world and processing the data.
- The system needs to present data easily on demand.
- The system must be able to handle heavy work loads.
- The configuration must not be limited in size.
- The system must be able to process a variety of applications.
- The system should not be limited to any particular field of activity.

- The system should be able to support transaction processing in open environments.
- The production system can be part of a global information system.
- Production systems can be at the core of task-driven solutions to overall large computing problems in organizations or departments.
- Production system environments can involve the use of PCs as terminals over PATHWORKS connections to OpenVMS servers (see Section 6.4.2).

7.1.1 OpenVMS Distributed Production Servers

OpenVMS VAX and OpenVMS Alpha systems provide the features needed for production systems of any size or configuration. OpenVMS configurations supply computing power, manageability, and security in predictable, stable environments. OpenVMS software ensures high application availability and data integrity. OpenVMS systems are dependable, exhibiting reliability, recoverability, and fault tolerance. These commercial-strength attributes of OpenVMS are summarized in Section 2.3.

OpenVMS systems are extremely well suited to function as production system servers in multivendor configurations involving distributed processing of applications. OpenVMS client/server capabilities in distributed environments are described in Section 6.4.

OpenVMS production servers support distributed production systems that involve high-integrity, fault-tolerant, and transaction-processing software. OpenVMS servers can be individual OpenVMS systems or VMScluster systems that can coexist in a heterogeneous network with servers from other vendors (for example, IBM servers). Client software can reside on OpenVMS systems and on any other vendor's system that can run distributed applications. Middleware client/server software services can facilitate integration of multivendor client/server configurations into a single distributed computing environment (see Section 6.3).

7.1.2 Providing Dependability in Production System Environments

OpenVMS meets the requirement of production systems for dependable software that ensures availability and data integrity. Dependability attributes of OpenVMS systems are discussed in Section 2.3.1. This section highlights the functions of particular OpenVMS integrated and optional software products that promote availability and data integrity in production systems.

Software for managing and monitoring production system environments is described in Section 7.1.3. Transaction-processing software that runs on OpenVMS systems is covered in Section 7.2 and Section 7.3.

7.1.2.1 Maintaining High Availability in Production System Environments

Software that guarantees high availability of OpenVMS systems in production system environments includes VMSclusters, Volume Shadowing, and DECamds, the Digital availability manager for distributed systems.

OpenVMS VMScluster systems form an ideal base for developing and running high-availability applications, such as transaction processing applications and server applications. VMScluster configurations provide for data sharing and independent failure characteristics (see Section 2.3.2).

Some of the major features of VMScluster systems that promote availability are the following:

- Ability to survive failure of any component
- Ability to be serviced without interruption to applications
- Functions as a single system but individual systems can leave the cluster while the cluster continues to operate
- Transparent sharing of resources and data
- Centralized data can be made available to large numbers of users
- Balanced work load across print and batch queues
- Can grow with an organization
- Flexible configurations, including large disaster-tolerant configurations (see Section 7.1.3.5)
- Support for the full range of OpenVMS systems on VAX and Alpha processors

OpenVMS Volume Shadowing is a system integrated software product that provides for high data availability for disk storage devices (see Section 2.3.2). Volume shadowing involves maintaining the same data on multiple disks, called a shadow set. Volume shadowing provides these availability features:

- Prevents data loss resulting from media deterioration, communication path failure, or through controller or device failure
- Provides continued access to data despite failures in the disk media, disk drive, disk controller, or OpenVMS host serving a shadow set member
- Prevents storage subsystem component failures from disrupting system or application operations
- Continues operation without interruption if one volume fails, automatically directing requests for data to an alternate volume
- Supports host-based shadowing, including shadowing of widely separated VMScluster systems connected by LANs
- Usable on any Digital Storage Architecture (DSA) disk and Digital SCSI disks on any VAX or Alpha processor or VMScluster system

Another tool used by OpenVMS system managers to improve availability is DECamds, a real-time monitoring, diagnostic, and correction tool. It collects and analyzes data from multiple nodes simultaneously, directing output to a centralized DECwindows display. DECamds detects resource availability and suggests corrective actions.

To improve system availability, DECamds provides the following features:

- Alerts users to resource availability problems, suggests paths of investigation, and recommends actions to improve availability
- Allows real-time intervention, even when remote nodes are hung
- Centralizes management of remote nodes over the LAN
- Provides an intuitive DECwindows graphical user interface that is easy to learn and use
- Provides the ability to modify cluster quorum

DECamds runs on a VAX or Alpha console, gathering data from remote VAX and Alpha nodes on which DECamds drivers reside. The application displays two windows: the event log window oriented toward corrective action, and the system overview window that offers user-directed data collection and display options.

7.1.2.2 Ensuring Data Integrity in Production Systems Environments

OpenVMS products that promote integrity on OpenVMS systems include RMS Journaling software and DECdtm services.

RMS Journaling software protects the data integrity of RMS files. RMS Journaling supports three types of journaling, each providing protection against a particular type of problem:

- After-image journaling that permits changes made to a data file to be redone: Designed to recover a file that was lost or corrupted through accidental deletion or device failure.
- Before-image journaling that returns a data file to the previous state, undoing changes made to the file: Useful if bad or erroneous data is entered into a file during an update session.
- Recovery-unit journaling that ensures a series of RMS operations called for by an application program are done in their entirety or not at all: Protects against inconsistent data due to the partial completion of transactions.

Recovery-unit journaling provides for transaction integrity. A transaction is a series of RMS record operations on one or more files and is viewed as an atomic operation, that is, all operations must be completed or none are (see Section 7.2).

The set of DECdtm services on the OpenVMS operating system (summarized in Section 3.1.6) provides basic operating system support for distributed transactions (as described in Section 7.2.3). DECdtm ensures transaction and database integrity during transaction processing through the use of the two-phase commit protocol, which ensures that all database resources commit to completing a transaction or none commit.

7.1.3 Managing and Monitoring Production System Environments

Production system work can be located in centralized data facilities or can be dispersed throughout a distributed environment. A distributed production system environment can involve a number of diverse systems used by a large number of people performing varied tasks. Managing complex computing environments often involves the use of multiple products and services.

OpenVMS supports a wide variety of optional software tools and utilities to assist users in managing complex distributed production system environments. OpenVMS capabilities for managing and providing security for large data centers and dispersed environments are summarized in Section 2.3.5. Additional capabilities are provided by software from independent software vendors.

Table 7–1 lists Digital-supplied software running on the OpenVMS system that can be used in managing complex production systems such as large data centers. The table covers optional software products and also the following software supplied with the OpenVMS operating system: Accounting utility, AUTOGEN, DECevent utility, Error Log utility, generic queues, Monitor utility, and OPCOM message routing.

The following sections provide additional information about some of the tools running on OpenVMS that are used to manage and monitor distributed production system environments.

Table 7–1 Software Used to Manage Complex Production Systems

Software Products	Function
	Accounting
OpenVMS Accounting	Facilitates user control
POLYCENTER Accounting Chargeback[1]	Provides basic user accounting and chargeback reports
	Application Management
ACMS and DECtp Desktop for ACMS	Provide recoverable applications using a transaction-processing monitor
DECADMIRE	Provides for generation of ACMS applications or DECforms and other vendor database applications, including client/server applications
	Application Debugging
DEC Performance and Coverage Analyzer (PCA)	Tunes and checks applications
	Archiving
Storage Library System (SLS)	Protects user data on other media
	File Shelving
POLYCENTER HSM for OpenVMS	Provides a file-shelving facility that can move the data portion of an inactive file to nearline or offline media transparently
	Capacity Planning
POLYCENTER Capacity Planner[1]	Provides a capacity-planning function to help the system manager analyze how changes in the configuration affect users' performance
	Event Messages
OPCOM message routing	Provides event notification
Error Log utility	Selectively reports the contents of an error log file containing events written by the OpenVMS system
DECevent Event Management utility‡	Selectively reports the contents of one or more event log files
POLYCENTER Console Manager	Consolidates system consoles and analyzes console messages for events
POLYCENTER System Watchdog	Provides for automated monitoring, detection, and reporting of hardware and software events, including events defined by a user application

[1] Requires POLYCENTER Performance Data Collector

‡Alpha specific

(continued on next page)

Table 7–1 (Cont.) Software Used to Manage Complex Production Systems

Software Products	Function
Job Scheduling	
POLYCENTER Scheduler	Schedules batch and print jobs
Generic queues	Uses VMScluster queues to feed queues on specific members across the cluster
Data Access	
DEC DB Integrator Gateways	A family of gateway products running on OpenVMS that allow users direct access to data in a wide variety of databases located anywhere on the network.
RMS Journaling for OpenVMS	Provides support for data recovery and transaction processing
Disk Striping	
Disk Striping Driver for OpenVMS	Spreads I/O load over multiple disk drives
Mirrored Disks	
OpenVMS Volume Shadowing (Phase II)	Helps system survive disk failure
Disk Array Management	
StorageWorks RAID Software for OpenVMS	Manages groups of disk drives as arrays
Disk Defragmentation	
POLYCENTER File Optimizer	Defragments disks by using the OpenVMS movefile subfunction
Forms Management	
DECforms	Separates form input from application management
Performance Management	
AUTOGEN command procedure	Optimizes system parameter settings based on usage
POLYCENTER Performance Advisor[1]	Identifies performance bottlenecks and recommends ways to fix problems
DECamds	Collects and analyzes data from multiple nodes, detects resource availability, and suggests corrective action.
DECram (in-memory disk/device)	Alleviates I/O bottlenecks
Monitor utility	Provides basic performance data

[1]Requires POLYCENTER Performance Data Collector

(continued on next page)

Table 7–1 (Cont.) Software Used to Manage Complex Production Systems

Software Products	Function
Software Installation	
POLYCENTER Software Installation utility	Supports a new technology that simplifies the distribution and management of software
LMF (License Management Facility) utility	Manages software licences
Remote System Management	
POLYCENTER Software Distribution Manager	Provides easy-to-use remote management of OpenVMS or Digital UNIX systems
System Configuration	
Remote Bridge Management System (RBMS)	Maintains a database of the setup of a system's complement of network servers and facilitates automatic loading of the servers upon system initialization
Terminal Server Manager (TSM)	Maintains a database of the setup of a system's complement of terminal servers and facilitates automatic loading of the servers upon system initialization
VAXsimPLUS †	Monitors device status and alerts the system manager of impending failure

†VAX specific

7.1.3.1 Storage Management Products

Optional products developed by Digital are used to manage mass storage devices and the data files stored on them in distributed production system environments. This section discusses examples of mass storage products that run on OpenVMS.

StorageWorks RAID Software for OpenVMS uses RAID (Redundant Array of Independent Disks) technology to manage groups of disk drives as arrays. RAID technology was developed as a means for lowering the cost of storage subsystems while maintaining high availability and improved performance over single disk drives. A RAID set is a collection of disks that uses various algorithms for availability and performance, depending on the level of RAID. There are six RAID levels; the highest performance level is 0. StorageWorks RAID Software supports RAID level 0 for enhanced I/O performance and RAID level 5 arrays for enhanced data availability on VAX and Alpha platforms. Both single-host and VMScluster systems are supported.

Reducing file fragmentation on OpenVMS systems is possible with the POLYCENTER File Optimizer for OpenVMS. This product provides the ability to reduce file fragmentation, schedule defragmentation jobs, optimize file placement on the disk, and select or exclude files. If a user accesses a file that is being defragmented, the file is released and defragmentation continues on another file. The defragmentation engine reads the disk's file structure and dynamically determines which files should be placed at which locations on the disk. Hot files (those files on the disk that are accessed most frequently, determined using the POLYCENTER Performance Solution tool) can be moved to the center of the disk. Files are moved to different locations on the disk through invocation of the OpenVMS atomic movefile system service.

The Storage Library System (SLS) automatically schedules backup operations for VMScluster systems, allowing the storage administrator to define which files should be backed up using what schedule. SLS submits the required backup jobs on schedule and tracks their progress. SLS automatically manages the usage, allocation, and tracking of magnetic tapes and write-once-read-many (WORM) optical disks.

POLYCENTER HSM for OpenVMS, a hierarchical storage management product, is a file-shelving facility that lowers the average cost of storing data that is rarely used. The shelving facility functions as an extension to the OpenVMS file system, allowing the user to move the data portion of an inactive file to nearline or offline media transparently. While the file is in the shelved state, the file remains visible in directories, and file attributes can be viewed and modified. When the file data is read or written, the data is automatically and transparently restored to the disk. File shelving permits users and applications to keep a large amount of data readily and easily accessible, without keeping it all on line and without having to enter explicit commands to move data between online and nearline or offline storage.

The Disk Striping Driver for OpenVMS combines multiple disks into one virtual disk or stripe set. Each stripe set, composed of multiple disk drives, appears to applications and utilities as a single pseudodevice capable of responding to I/O operations defined for ordinary disk drives. Through the use of parallel I/O technology, the Disk Striping Driver can spread a file over several disk drives and read or write it over several channels simultaneously. Use of a stripe set increases I/O performance over the use of a single disk drive or a set of unstriped disks. This product solves I/O performance problems and is useful in load balancing on OpenVMS systems.

Disk striping is RAID level 0, the highest performance RAID. Volume shadowing is RAID 1. Striping plus shadowing offers the highest performance and the highest availability of any RAID implementations.

7.1.3.2 DECram for OpenVMS Device Driver

DECram for OpenVMS is an integrated disk device driver that enables OpenVMS system managers to create pseudodisks in the VAX or Alpha system main memory to improve I/O performance. The OpenVMS operating system can use standard OpenVMS disk I/O operations to access data on a DECram disk, but at a much greater access rate than for standard hardware disks. The DECram disk, resident in main memory, is accessed through the OpenVMS file system in the same way that physical disks are accessed; no changes are required to applications or system software. DECram disks may be used to store frequently needed read-only data (such as commonly used image files) or to hold temporary or scratch files for applications. The DECram disk information is volatile; it is not saved when the system shuts down or crashes.

7.1.3.3 Performance Management Tools

The following optional tools that run on OpenVMS VAX and OpenVMS Alpha systems help manage performance:

- POLYCENTER Performance Solution products provide capabilities for performance measurement and capacity management. These products increase productivity by automating data collection, utilizing artificial intelligence (AI) techniques for performance-tuning recommendations, and enabling interactive modeling for capacity planning. Extensive graphing and reporting facilities supply essential details of system activity. The information provided by these products helps to optimize the use of current resources as

well as to plan more accurately for growth. POLYCENTER Performance Solution enables data center managers to use a proactive approach to planning.

POLYCENTER Performance Solution is an integrated set of OpenVMS layered products, as follows:

- The POLYCENTER Performance Data Collector gathers and manages OpenVMS system data; it must be installed on each system.
- The POLYCENTER Performance Advisor provides performance analysis, graphing, and reporting capabilities; it analyzes data collected using expert system techniques and generates a statistical overview of actual system performance.
- The POLYCENTER Capacity Planner determines system performance for various work loads and configurations, using data collected to perform capacity planning exercises.
- The POLYCENTER Accounting Chargeback software allocates charges for resource usage and generates reports that can be used as an itemized bill or a general resource utilization report.

See Section 7.1.2.1 for a description of DECamds, which provides real-time observations of system resources.

7.1.3.4 Optional VMScluster System Management Software

OpenVMS supports optional software products that provide management capabilities and services for complex distributed environments. These tools can run on VMScluster systems.

OpenVMS can manage other systems remotely over the network. The POLYCENTER Software Distribution Manager (previously known as the Remote System Manager), which runs on OpenVMS VAX and OpenVMS Alpha, is an optional network product that permits a system manager to manage a number of OpenVMS systems and UNIX based systems connected to a DECnet network. The system manager operates from a management server to install software remotely and schedule or initiate backups for the remote client's system. The product permits the manager to manage more systems and offloads system management tasks from users on individual systems. The POLYCENTER Software Distribution Manager supports client systems ranging from single workstations to VMScluster members to PC LAN servers.

POLYCENTER Scheduler for OpenVMS automatically schedules, executes, and monitors tasks such as backup, file maintenance, and production jobs. It also supports complicated, repetitive batch applications such as payroll, manufacturing resource planning, and financial consolidations.

The POLYCENTER System Watchdog Agent (WDA) and Consolidator (WDC) are optional products that provide automated detection, notification, and correction of system faults and problems. The System Watchdog monitors network, systems, subsystems, processes, and user-defined external events, detecting and reporting changes in resources before serious problems occur, in order to optimize system availability for users. The POLYCENTER System Watchdog (which replaces the Data Center Monitor [DCM] product) is part of the integrated set of system management tools provided by the POLYCENTER Solution (see Section 6.3.6).

The POLYCENTER Console Manager consolidates the console management of distributed heterogeneous equipment. One host can manage multiple computing resources over standard lines. The POLYCENTER Console Manager notifies personnel of critical systems events and/or activates corrective action routines.

VAXsimPLUS is a knowledge-based predictive maintenance tool that continually monitors all system devices to detect problems before they result in down time. VAXsimPLUS predicts impending failures and provides dynamic disk substitution and restoration across DECnet, VMScluster, and other systems.

7.1.3.5 Business Recovery Server

The optional Business Recovery Server (discussed in Section 2.3.5) provides very high levels of system availability and data protection. The server enables management of widely distributed VMSclusters that are connected in disaster-tolerant configurations. Integrated software permits CPUs in multiple data centers to be combined into a single manageable VMScluster system. The menu-driven installation procedure automatically installs various Business Recovery Server components and the correct versions of required layered products. In addition, menu-driven configuration software simplifies such activities as adding a node or configuring network devices.

The Business Recovery Server consists of data centers that are located a safe distance apart. Each data center is controlled by an Operator Management Station (OMS), which makes it possible to monitor and manage these physically separate data centers across great distances. Sites at a distance of up to 25 miles apart can be connected using the high-speed FDDI fiber-optic interconnect; sites at a distance of up to 500 miles apart can be connected using the FDDI along with ATM and T3 long-distance communication services.

The OMS node is unique to the Business Recovery Server and plays a central role in cluster management and disaster recovery. The OMS software in a Business Recovery Server cluster provides even greater disaster tolerance than the highly reliable and dependable VMSclusters that do not use Business Recovery Servers. Configured with the recommended hardware, the OMS software:

- Monitors system traffic for any irregularities and copes with a wide range of possible malfunctions
- Enables management of both locations from either OMS, with the option to change control from one site to another on demand
- Automatically transfers control to the redundant OMS at the secondary site if disaster strikes the primary site (the site with system and network management control)

Multivendor Customer Services offers consultation packages that assist customers in configuring highly available, disaster-tolerant Business Recovery Server clusters.

7.2 Transaction Processing in Multivendor Environments

Transaction-processing applications are typically described as critical or ‘‘bet your business’’ applications, which consequently require the highest degree of availability, security, and integrity. These applications are usually high-volume, online applications that process transactions in real time and reflect the most current state of the business.

Transaction processing (TP) is the implementation of computer transactions, which supply the mechanism for accomplishing a business transaction. A computer transaction is a series of record operations made on one or more files. Several computer transactions are often required to implement a single business transaction (such as making a travel reservation or making an electronic funds transfer from one bank account to another).

The transaction application consists of one or more application programs that receive the input data and initiate the required transaction. Digital transaction-processing software provides the underlying integrity for the application, monitoring the transaction through completion or, in the event of failure, canceling every step.

7.2.1 Distributed Transaction-Processing Systems

Traditionally, transaction-processing applications have been developed to run on a single computer such as a large mainframe system. The trend is toward distributed client/server transaction-processing systems, in which an application runs on a set of systems linked in a network.

Transaction-processing solutions provided by Digital permit transactions to be executed at multiple sites and combined with data stored over widely distributed nodes or to be centralized on a single computer. Distributing transaction-processing applications across multiple systems provides the following benefits:

- Local data can be kept close to users.
- Processing can be offloaded from more expensive mainframe systems.
- Higher overall throughput can be achieved.
- Communication costs can be cut.
- Scalable solutions can be configured to meet changing needs.

Transaction-processing software that runs on the OpenVMS system involves TP system functions that are separated into discrete components with specified interfaces between them. Individual components can be changed without affecting the others and can be redistributed without impacting the applications. Users can choose to mix and match components from various vendors as long as the components comply with the standard interfaces.

High-level software systems that run on OpenVMS systems[2] provide capabilities needed for the core applications of commercial customers:

- DECtp is a transaction-processing system that provides control and management of TP applications involving multiple users accessing a common database. The DECtp system is based on the ACMS transaction-processing monitor (see Section 7.2.2), DECforms (see Section 5.4.1), and relational database management system software supplied by other vendors, along with other software development tools. For multivendor desktop systems, the DECtp Desktop for ACMS product (described in Section 6.4.6) is available.
- DEC Reliable Transaction Router (RTR) is a distributed software message routing system that supports TP applications (see Section 7.2.4). DEC RTR permits data to be partitioned over multiple nodes and disks; the location of the data is transparent to the application.

[2] Refer to the applicable Software Product Description for information about support for each product.

7.2.2 Distributed Transaction-Processing Monitors

A transaction-processing monitor is a collection of components bundled together in a tightly integrated package to facilitate application development and provide high performance and dependability in the production environment.

ACMS is an application control and management system for developing, controlling, and maintaining transaction-processing applications. It consists of front-end clients that collect multiple transactions for the back-end servers that execute the database I/O. ACMS provides operating system interfaces, standard menu-driven functions, terminal and file I/O services, and other commonly needed services for building transaction-processing applications, using most OpenVMS higher level languages. It is designed for high-productivity application development.

ACMS is a highly reliable TP system that includes dependability features to keep applications up and running at optimal efficiency, recoverability, and durability. Some of the major dependability features are:

- Queue management
- Load balancing and scheduling
- Application scheduling
- Deferred transaction scheduling
- Transaction routing
- Security
- Menu subsystems
- Message recovery
- Debugging support
- Interfaces to nonstandard or other-vendor devices

ACMS is fully integrated with DECdtm services, DECforms, and database software supplied by other vendors. This group of products is part of the Digital software development environment for providing development, run-time, and management features for transaction-processing applications. The high-availability features inherent in these software products combine to provide smooth transitions when problems or failures occur. Combining ACMS with other vendor relational database management software improves the system's ability to resist most failures.

Features of ACMS that provide for fault management or recovery from failures that can occur during run time include:

- Automatic front-end terminal failover
- Use of queues to capture user requests
- Balancing of process pools
- ACMS application failover

ACMS works with DECdtm services provided by the OpenVMS operating system to ensure transaction commits, aborts, and recoveries when failures occur. DECdtm services maintain database integrity through a two-phase commit transaction protocol (described in Section 7.2.3) to coordinate the flow of transaction requests and make sure that all of the operations of a transaction are

performed, or that none of them are performed. If all operations are performed, the transaction is committed, and the database is in a consistent state.

The Digital forms management package, DECforms (described in Section 5.4.1), integrates with ACMS to provide forms-processing capabilities in transaction-processing environments. Data management is provided by OpenVMS RMS and database products supplied by other vendors.

7.2.3 Transaction-Processing Support by DECdtm Services on OpenVMS Systems

DECdtm services available on every OpenVMS VAX and OpenVMS Alpha operating system ensure transaction and database integrity when used with multiple transaction-processing monitors and database products in local and remote locations in distributed processing environments. Applications developed to handle distributed transactions can include calls to Digital data management products.

DECdtm services comprise several components, including a transaction manager, interfaces to system services, logging and recovery services, and communication services.

The DECdtm transaction manager (TM) supports services, issued from the transaction-processing applications, to start, end, or abort transactions. The TM coordinates the action of a distributed transaction by sending instructions to resource managers (such as RMS Journaling and other vendor database products) about how to complete the transaction. Each OpenVMS system ordinarily includes one DECdtm object, containing the TM for transactions initiated from that node.

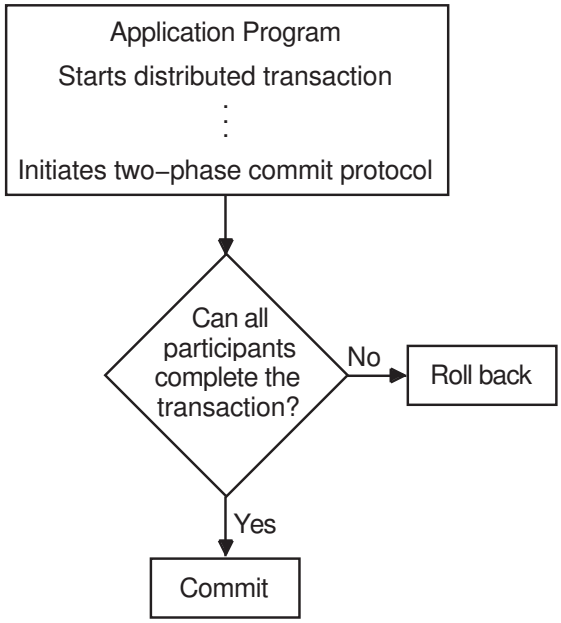
The DECdtm log manager provides a mechanism for storing a permanent record of the execution of distributed transactions. If there is a failure, a resource or transaction manager can read the log file to determine the state of the transaction at the time of the failure.

DECdtm implements a two-phase commit protocol to guarantee atomic transaction processing (see Figure 7–1). The two-phase commit protocol is a handshaking procedure in which all participants in a distributed transaction first agree to commit and then, on a signal from a coordinator, actually do commit. All resources commit, or none of the resources commit; no operation is ever partially applied to a database. The commitment protocol permits the development of distributed TP applications that can survive the failure of individual CPUs and mass storage subsystems.

DECdtm provides the following data integrity features:

- Ensures consistent execution of distributed transactions on the OpenVMS operating system.
- Ensures database integrity by guaranteeing that transactions spanning multiple RMS files and databases in local and remote locations finish as a single unit or not at all.
- Use of the two-phase commit protocol permits a large relational database to be divided into several smaller databases and allows applications to access several different databases, without compromising the integrity or consistency of the data.

Figure 7–1 Two-Phase Commit Protocol for a Distributed Transaction



ZK–1772A–GE

- Supported by transaction-processing monitors (ACMS), RMS Journaling, and multiple database products supplied by other vendors.

7.2.4 Other Distributed Transaction-Processing Products

In addition to the DECtp system, the transaction-processing capability includes the DEC Reliable Transaction Router (RTR) and, optionally, DSM, both of which are available on OpenVMS VAX and OpenVMS Alpha.

DEC RTR is a distributed, fault-tolerant message-routing system that supports several types of distributed applications, including TP. DEC RTR is designed to guarantee the delivery of a transaction message from a client to a server across a network for execution on distributed VAX or OpenVMS Alpha systems anywhere in the world. Reliability is provided by software fault tolerance through redundant paths and redundant systems, if necessary. DEC RTR is suitable for customers with multiple geographic locations, large systems, the need to integrate distributed applications, and the requirement for the highest levels of reliability, availability, fault tolerance, and high-performance throughput. DEC RTR is used for production systems in large, distributed environments (for example, trading or money-dealing systems and telecommunications systems). The global application integration and software fault-tolerant protection of DEC RTR ensures that server applications anywhere in the network are alway available to clients.

The optional DSM system, Digital's standard MUMPS software, can run on systems to build and support large, complex, distributed systems for many industries such as health care, banking, government, transportation, shipping, and manufacturing. DSM is a high-performance system, consisting of an interactive, interpretive programming language, database manager, and forms manager, that can be used to build, test, install, and maintain an application. DSM is an open, callable system supporting interoperability with other layered systems.

7.3 Database Processing in Multivendor Environments

OpenVMS systems and optional products support the strategy of making information on any system in the organization transparently available to any authorized user or application, regardless of which computers are used. The OpenVMS environment includes tools and software useful in developing and managing an open information network across an entire enterprise.

Information systems for general office computing, commercial applications, online transaction-processing, or end-user applications are based on the database management system. OpenVMS VAX and OpenVMS Alpha systems provide support for database management systems supplied by Digital (RMS) and other vendors (for example, Oracle Rdb and Oracle CODASYL DBMS). (RMS, which is integrated in the OpenVMS operating system, is described in Section 4.8.) A number of database tools support these database management systems in distributed environments.

Digital supports a family of data access and integration products that permit users to access data stored in multiple databases with multiple formats as though the data were in a single logical database (see Section 7.3.2).

7.3.1 Database Tools Used in a Distributed Environment

Database tools that support database management include:

- DECquery for MS–DOS and DECquery for Windows: Permits PC users to develop data queries without knowledge of SQL
- DATATRIEVE: A structured, nonprocedural language that provides selective data retrieval, sorting, formatting, updating, and report generation for Rdb, DBMS, and RMS formats
- Compound document services: Support the creation, display, processing, and interchange of different data types (including text, graphics, images, voice, tables, and multimedia) on different computing platforms. CDA services provide a consistent way for developers to build applications that can create, store, view, and share revisable compound documents among applications on multiple computing systems. (The CDA Viewer available with the DECwindows Motif user interface is described in Section 5.2.2.)

Database software from other vendors (such as Oracle and Sybase) are also used with OpenVMS servers in applications such as transaction-processing applications.

7.3.2 Database Interoperability Software

Digital provides integrated products that permit data sharing in heterogeneous environments. Users can have a single, consistent interface to directly access major data sources in an organization. Many types of applications from multiple vendors on various platforms can exchange data and other functions across a network.

The Digital family of access and integration products provides a complete set of software to promote data access and interoperability. These products are especially suited for production-level, enterprisewide solutions where flexibility, performance, data integrity, and the handling of large amounts of data are required.

DEC DB Integrator provides users with the capability to transparently access data stored in multiple homogeneous or heterogeneous databases from SQL-based tools and applications that reside on multiple platforms (including MS–DOS, Windows, Macintosh, Sun, and UNIX systems as well as OpenVMS VAX and OpenVMS Alpha systems). DB Integrator enhances interoperability products from Digital and other vendors by providing additional transparency when accessing data stored in multiple databases, in different locations, and with multiple formats. To the user, the data appears as if it were stored in one logical database.

DB Integrator works in conjunction with DEC DB Integrator Gateway to provide integration across different combinations of databases as well as relational and non-relational data sources.

Digital's database gateways allow users direct, transparent access to data located in a very wide variety of databases. Using the DEC DB Integrator Gateways, users with a desktop tool or application can retrieve or update data from a data source located anywhere on the network. DB gateway products permit access to OpenVMS RMS, DEC DSM, Oracle, DB2, Sybase, and EDA/SQL[3] databases and to proprietary data sources through custom drivers.

DEC Data Distributor manages the automated distribution of data among multiple nodes, using schedulable data transfers. The source database can be a relational database or one of the DB Integrator Gateway databases. The target can be a relational database or a DB gateway database. Data can be extracted from very large production databases and can be automatically combined from multiple databases into one.

DB Integrator is the data integration component of Digital's client/server software configurations. DB Integrator is incorporated with networking, hardware, and software tools and services to provide a preconfigured data integration solution running on an OpenVMS server. This solution, called the DEC ACCESSWORKS product family (described in Section 6.4.6), allows application programs running on various desktop computers to access information in multiple database systems over PATHWORKS network connections. Users can access data transparently from a variety of database management systems, including EDA-supported databases.

[3] More than 50 databases accessible through the Information Builders, Inc. (IBI) Enterprise Data Access/Structured Query Language (EDA/SQL) application programming interface.

A

OpenVMS Support for Standards

The OpenVMS operating system and related optional products support industry and international standards and specifications. This appendix provides summary information about OpenVMS conformance and compliance with standards and specifications adopted by recognized standards bodies, including the following:

- American National Standards Institute (ANSI)
  - Forms Interface Management System (FIMS) [proposed ANSI/ISO standard]
- Institute of Electrical and Electronics Engineers (IEEE)
  - Portable Operating Systems Environment Interface (POSIX)
- International Organization for Standardization (ISO)
- International Telegraph and Telephone Consultative Committee (CCITT) [recommendations]
- Internet Engineering Task Force (IETF)
- National Institute of Standards and Technology (NIST)
  - U.S. Federal Information Processing Standards (FIPS)
- Object Management Group (OMG)
- Open Software Foundation (OSF):
  - OSF Application Environment Services (AES)
  - OSF Distributed Computing Environment (DCE)
- U.S. National Computer Security Center (NCSC)
  - Department of Defense System Trusted Computer System Evaluation Criteria (TCSEC)
- X/Open Portability Guide Issue 3 (XPG3) and Issue 4 (XPG4)
- X Window System (Version 11)

A summary of the types of standards, draft standards, and specifications supported in major areas of OpenVMS capability appears in Table A–1.

Table A–1 OpenVMS Support for Industry and International Standards and Specifications

Technical Area	Type of Standard or Specification
Operating system information interchange	ANSI, FIPS, ISO
Operating system interfaces	IEEE POSIX 1003.1, 1003.1a (draft), 1003.1b
	XPG3 BASE
	XPG4 BASE profile
User interfaces	X Window System
	OSF/Motif
	POSIX 1003.2
	XPG3 BASE
	XPG4 BASE profile
Small systems interface	ANSI SCSI-2
Local area networks (Ethernet and FDDI)	IEEE 802/ISO 8802 (CSMA/CD)
Security	TCSEC from NCSC (OpenVMS VAX is evaluated at C2)
Management	OSF DCE
	ISO (CMIP)
	ISO 9660
Data interchange	ISO
	ANSI X.12 (EDI)
	ISO SGML
Languages	ANSI/MIL-STD/ISO Ada
	ANSI/ISO BASIC
	ANSI/ISO C
	C++
	ANSI/ISO COBOL
	ANSI/ISO FORTRAN, FIPS
	ANSI/ISO Pascal
Networking and communication	ISO standards (OSI model)
	X.400 CCITT/ISO
	X.500 CCITT/ISO
	X.25 CCITT
	ISO FTAM
	TCP/IP
	NFS
	IETF RFC 822
	IETF RFC 1006, 1006 Plus
	PostScript
	Display PostScript

(continued on next page)

Table A–1 (Cont.) OpenVMS Support for Industry and International Standards and Specifications

Technical Area	Type of Standard or Specification
Distributed applications	OSF DCE
	OMG
Graphics interfaces	ISO GKS
	ISO GKS 3–D
	ISO PHIGS
Forms interfaces	ANSI/ISO FIMS (proposed)

Index

A

Access
  database, 6–24, 7–15
  OpenVMS systems, 5–1
Access control lists
  See ACLs
Access modes, 3–1
ACCESSWORKS, 6–24
Accounting utility (ACCOUNTING), 3–12
ACLs (access control lists), 3–18
  protection mechanism, 5–1
ACMS, 7–5
  dependability, 2–13, 7–12
  failure resistance, 7–12
  fault management, 7–12
  software, 7–12
  use with DECtp Desktop for ACMS, 6–25
Ada, 4–5
Addressing
  32-bit, 3–2
  64-bit, 3–2
  virtual, 3–2
AI (artificial intelligence), 4–6
  performance-tuning techniques, 7–8
Alias node identifier, 3–21
ALL–IN–1
  accessing the environment, 5–2
  user environments, 5–8
Alpha architecture, 1–13
Alpha platforms, 1–3
Alpha processors, 1–1, 1–14
American National Standards Institute
  See ANSI
ANALYZE/AUDIT
  See Audit Analysis utility
ANALYZE/ERROR_LOG command, 3–12
Anonymous FTP, 6–7
ANSI (American National Standards Institute), 2–2, 6–11, A–1
APL, 4–5
AppleShare, 6–21
AppleTalk, 1–9, 6–18
  protocols, 6–21
Applications
  components, 4–2
  DECwindows Motif, 5–6
  developing, 4–1
  distributed, 2–6
  graphical user interface, 4–18
  object-oriented, 6–15
  OpenVMS, 1–2
  portable, 2–3, 4–3, 5–1
  POSIX, 2–4, 3–4, 4–3
  real-time, 4–18
  three-dimensional graphics support, 6–16
  transaction-processing, 2–13
Architectures
  Alpha, 1–13
  CISC, 1–13
  RISC, 1–13
  VAX, 1–13
Artificial intelligence
  See AI
Assemblers, 4–5
ASTs (asynchronous system traps), 4–11
Asynchronous system traps
  See ASTs
ATM communications services, 1–8
Audit Analysis utility (ANALYZE/AUDIT), 3–19
Auditing subsystem, using for security, 3–17
Audit trails, security, 3–18
Authorize utility (AUTHORIZE), 3–10, 3–19
AUTOGEN command procedure, 3–8, 3–12
Availability, 7–2
  software tools, 2–12
Availability manager
  See DECamds

B

Backups
  of PC disks, 6–24
  on Alpha systems, 3–11
  on VAX systems, 3–11
  using SLS, 7–8
  using standalone BACKUP, 3–11
Backup utility (BACKUP), 3–11

BASIC
DEC BASIC for OpenVMS Alpha, 4–5
VAX BASIC, 4–5
Batch mode, 1–3
Batch operations, 3–13
BLISS-32, 4–5
BLISS-64, 4–5
Bookreader, 5–6
Booting, 3–8
remote services, 6–24
Buses
open, 1–5
storage, 1–5
Business Recovery Server, 2–11, 2–13, 7–10
Buttons, DECwindows Motif, 5–5

C

C
DEC C for OpenVMS Alpha, 4–5
VAX C, 4–5
C++, 4–5
Callable system routines, 4–9
Calling standard, 4–2
Captive accounts, 5–2
CCITT (International Telegraph and Telephone Consultative Committee), A–1
recommendations, 6–3
CDA compound document architecture, 7–15
CDA Viewer, 5–6, 5–7
CD–ROMs (compact discs read-only memory), 1–6, 6–9
CDS (Cell Directory Server), 2–9, 6–14
CDU
See Command Definition Utility
Cell Directory Server
See CDS
Central processing units
See CPUs
CI computer interconnect, 1–6
CISC
architecture, 1–13
Clients, 2–6, 6–17
PATHWORKS, 6–18
to OpenVMS distributed servers, 7–2
Client/server computing, 1–5, 2–6
in multivendor environments, 6–17
in production systems environments, 7–2
middleware support for, 6–10
Cluster-accessible disks, 3–22
Cluster-accessible tapes, 3–22
CMS, 4–9, 6–16
COBOL
DEC COBOL, 4–6
Code Management System
See CMS
Command Definition utility (CDU), 4–8, 4–17
Command procedures, 5–4
Common Object Request Broker Architecture
See CORBA
Communication service
middleware, 6–11
Communications services
ATM, 1–8
DS3, 1–8
T3, 1–8
Compact discs
See CD–ROMs
Compilers, 4–5
IDL, 6–14
Compound document architecture
See CDA
Computation services
middleware, 6–11
Computer interconnect
See CI
Computing styles, 1–3
Configurations
hardware, 1–7
standalone, 1–7
VAXcluster, 1–7
VMScluster, 1–7
Connection manager, VMScluster, 3–21
Container file systems, POSIX, 4–19, 5–10
Control services
middleware, 6–11
CORBA (Common Object Request Broker Architecture), 6–15
Core services, OpenVMS, 3–1, 3–4
CPUs (central processing units), 1–4, 1–7

D

Databases
access, 6–24, 7–15
distributed tools, 7–15
interoperability, 7–15
management systems, 7–15
Data centers
Business Recovery Server, 2–13, 7–10
managing, 2–13, 7–4
Data Distributor, 7–16
Data integrity
DECdtm, 7–13
production systems, 7–4
software tools, 2–12, 7–4
Data management, 4–19
DATATRIEVE, 7–15
DB Integrator, 7–16
DB Integrator Gateway, 7–6

DCE (Distributed Computing Environment)
product family, 2–9
services and interfaces, 2–9
software, 6–13
standards, 2–2, A–1
support for network transports, 6–15
DCE Application Development Kit for OpenVMS, 2–9, 6–14
DCE CDS
See CDS
DCE cells, 6–14
DCE DTS
See DTS
DCE Remote Procedure Call
See RPC
DCE Run-Time Services for OpenVMS, 2–9, 6–14
DCE Security Server, 2–9, 6–14
DCE Threads Service, 2–9, 6–14
DCL (DIGITAL Command Language), 3–4, 5–2
command procedures, 5–2, 5–4
file commands, 5–10
file-handling commands, 5–9
HELP command, 5–4
setting logical names, 5–3
SHOW command, 3–12
task commands, 5–3
Debuggers, 4–7
DEC ACCESSWORKS
See ACCESSWORKS
DEC Ada
See Ada
DECADMIRE, 6–16, 7–5
DECamds, 2–12, 7–3, 7–6
DEC BASIC
See BASIC
DEC BLISS-32
See BLISS-32
DEC BLISS-64
See BLISS-64
DEC C
See C
DEC C++
See C++
DEC COBOL
See COBOL
DEC Code Management System
See CMS
DEC Data Distributor
See Data Distributor
DEC DB Integrator
See DB Integrator
DEC DB Integrator Gateway
See DB Integrator Gateway
DECdfs, 2–10, 6–7
DECdns, 2–10, 6–4
DECdtm services
data integrity, 7–4
log manager, 7–13
software components, 3–7
support for transaction processing, 2–12, 4–12, 7–13
transaction manager (TM), 7–13
DECdts, 2–10, 6–4
DEC/EDI, 6–5
DECevent Event Management utility, 7–5
DECforms, 2–13, 7–6
user interfaces, 5–8
using in transaction processing applications, 7–13
DEC Fortran
See Fortran
DEC Language-Sensitive Editor/Source Code Analyzer
See LSE/SCA
DEClinks, 4–18, 5–7
DEC MailWorks
See MailWorks for OpenVMS
DECmigrate for OpenVMS AXP, 1–14, 4–6
DEC Module Management System
See MMS
DECnet
compatibility, 2–10, 2–16, 6–3
networking products, 1–9
networks, 6–2
network transport, 6–21
security, 6–8
user applications, 6–6
DECnet/OSI
software, 6–3
DECnet/OSI for OpenVMS, 1–9
features, 6–4
layers, 6–3
NCL command language, 6–5
network management, 2–16, 6–4, 6–5
open networking, 2–10
OSI model, 2–5
standards supported, 6–3
DECnet for OpenVMS, 1–9, 2–10
NCP command language, 6–5
network management, 2–16
network management tasks, 6–5
Phase IV, 6–3
Phase IV protocols, 1–9
DECNIS network integration server, 1–11, 6–9
DEC Notes
See Notes
DEComni, 6–5
DEComni MMS, 6–5
DEC OPS5
See OPS5

DEC Pascal
See Pascal
DEC PL/I
See PL/I
DECplan, 6–16
DECquery
for MS–DOS, 7–15
for Windows, 7–15
DECram, 2–14, 7–6
device drivers, 7–8
DEC Reliable Transaction Router
See RTR
DECrpc, 2–10
DECservers, 6–9
DECset, 4–18, 6–16
DECsound, 5–6
DECterm, 5–6, 5–7
DEC text-processing utility
See DECTPU
DECthreads, 3–3, 4–9
DECtp, 7–11
DECtp Desktop for ACMS, 6–25, 7–5, 7–11
DECTPU, 4–4
EVE, 5–12
DEC WANrouter
See WANrouter
DECwindows Motif
applications, 5–6
buttons, 5–5
client/server software, 2–10, 6–18
components, 5–5
DECterm window, 5–3
dialog boxes, 5–5
distributed features, 2–10
FileView, 5–5, 5–9
graphical user interface, 1–3, 5–2, 5–4
Help menu, 5–5
icons, 5–5
menus, 5–5
portable user interfaces, 2–4
programming libraries and tools, 4–18
Session Manager, 5–5
user environments, 5–5
Workspace, 5–5
Delta/XDelta Debugger (DELTA/XDELTA), 4–7
Dependability, 7–2
OpenVMS systems, 2–11
DESNC (Digital Ethernet Secure Network Controller), 6–8
Device drivers, 1–7
OpenVMS, 3–3
user-written, 3–3
Dialog boxes, DECwindows Motif, 5–5
DIBOL
VAX DIBOL, 4–6
DIGITAL Command Language
See DCL
Digital Extended Math Library for OpenVMS, 4–11
Digital GKS
See GKS
Digital Network Architecture
See DNA
Digital Open3D for OpenVMS Alpha
See Open3D
Digital PHIGS
See PHIGS
Digital standard MUMPS
See DSM
DIGITAL Standard Runoff
See DSR
Digital Storage Architecture
See DSA
Digital storage system interconnect
See DSSI
Digital TCP/IP Services for OpenVMS
See TCP/IP
Digital VTX
See VTX
Directories
OpenVMS, 5–9
user, 5–9
Disaster-tolerant systems, 2–11, 7–10
Disks
cluster-accessible, 3–22
defragmenting, 7–6
Disk servers, VMScluster, 3–21, 6–18
Disk striping, 2–14, 7–8
Disk Striping Driver for OpenVMS, 2–14, 7–6, 7–8
Distributed computing, 2–6
Distributed Computing Environment
See DCE
Distributed environments, 2–6, 6–2
heterogeneous networks, 6–2
Distributed file system, VMScluster, 3–21
Distributed lock manager, VMScluster, 3–21
Distributed processing, 1–5, 2–1
transactions, 7–11
Distributed production systems, 7–1
Distributed Time Service
See DTS
Distributed transactions, 2–13, 3–7, 7–13
DNA protocols, 1–9, 6–3
DNS/BIND, 6–4
DOCUMENT, 5–12
DS3 communications services, 1–8
DSA (Digital Storage Architecture)
disks, 3–22
tapes, 3–22

DSM, 7–14
DSR (DIGITAL Standard Runoff), 5–12
DSSI, 1–6
DTS (Distributed Time Service), 6–14
Dual-architecture VMScluster systems, 1–7

E

Editors, 4–4
  ed (POSIX), 5–12
  EDT, 4–4, 5–12
  EVE, 4–4, 5–12
  TECO, 4–5
  text file, 5–12
  vi (POSIX), 4–5, 5–12
Electronic conferencing
  See Notes
Encryption, 2–14, 3–17, 6–8
Environments
  ALL–IN–1, 5–2, 5–8
  client/server, 2–6, 6–17
  common code, 4–2
  common language, 4–2
  common programming, 4–1
  computing, 2–1
  DECwindows Motif user, 5–5
  distributed, 2–6, 6–1, 6–2
  hyperinformation, 4–18, 5–7
  multivendor, 2–5, 6–2
  open system, 2–1
  OpenVMS, 5–2, 6–1
  PATHWORKS, 6–18
  POSIX, 5–2, 5–7
  production systems, 2–1, 7–1
  security, 3–17
  software development, 6–15
  standalone, 6–1
  VMScluster, 1–7
Error Log utility (ERROR LOG), 7–5
Ethernet
  communications interconnects, 1–8
  LAN adapter, 1–6
  LAN technology, 6–18
  networks, 1–11
EVE (Extensible Versatile Editor), 4–4, 5–12
Event, 3–3
Event flags, 3–4, 4–11
EXCHANGE/NETWORK command, 5–11
Execution queues, 3–13
Extensible Versatile Editor
  See EVE

F

Fault tolerance, 2–11
FDDI (Fiber Distributed Data Interface), 1–6
  communications interconnects, 1–8, 1–11
Federal Information Processing Standards
  See FIPS
Fiber Distributed Data Interface
  See FDDI
File Optimizer, 2–14
  defragmenting disks, 7–6
  reducing file fragmentation, 7–7
Files
  container systems, 5–10
  DCL commands for, 5–9, 5–10
  defragmenting, 7–7
  log, 3–12
  OpenVMS, 5–9
  optimizing, 7–6
  POSIX, 5–9, 5–10
  POSIX commands for, 5–12
  POSIX utilities for, 5–12
  protecting, 3–17
  RMS, 4–19
  sharing, 1–7
  specifying OpenVMS, 5–9
  specifying POSIX, 5–10
  text editor, 5–12
  transferring, 5–11, 6–5
  utilities for, 5–10
File shelving, 2–14, 7–8
File Transfer, Access, and Management Service
  See FTAM
FileView
  DECwindows Motif, 5–5
  graphical interface, 5–9
FIMS (Forms Interface Management System), 5–8, A–1
FIPS (Federal Information Processing Standards), 2–2, A–1
Forms
  electronic, 5–8
  standards, 5–8, A–3
Forms Interface Management System
  See FIMS
Forté, 6–16
Fortran
  DEC Fortran 90, 4–6
  DEC Fortran for OpenVMS Alpha, 4–6
  DEC Fortran for OpenVMS VAX, 4–6
FTAM (File Transfer, Access, and Management Service), 6–5

G

Generic queues, 3–13, 7–6
GIGAswitch device, 6–9
GKS, 6–16
Graphical interfaces
  creating, 4–18
Graphics
  applications support for, 6–16
  standards, A–3

H

Hardware configurations
  standalone, 1–7
  VMScluster, 1–8
Help menu, DECwindows Motif, 5–5
Help Message utility (MSGHLP), 5–4
Help system, 5–4
High-performance sort, 4–13, 5–11
HSC hierarchical storage controller, 1–8, 3–22
Hyperapplications, 5–7
Hyperinformation environment, 4–18, 5–7

I

I/O performance
  using DECram, 7–8
  virtual I/O cache, 3–13
I/O subsystem, 3–1, 3–3
  connections, 1–5
I/O system services, 4–12
IBM SNA interconnect products, 1–11, 6–5
Icons, DECwindows Motif, 5–5
IDL (Interface Definition Language), 6–14
  compiler, 6–14
IEEE (Institute of Electrical and Electronics Engineers), 6–11, A–1
  definition of open system, 2–1
  POSIX standards, 2–2
  standards, 6–3
IETF (Internet Engineering Task Force), 6–11, A–1
  Request for Calls, 5–13
  Request for Comments, 2–2, 6–3
Images, 3–1, 5–2
Information handling on OpenVMS systems, 5–9
Information management, 7–15
Information services
  middleware, 6–11
InfoServer, 1–6, 3–14, 6–9
Installation
  OpenVMS Alpha systems, 3–8
  OpenVMS VAX systems, 3–8
Institute of Electrical and Electronics Engineers
  See IEEE
Integration
  data, 7–16
  multivendor, 6–9
Interactive mode, 1–3
Interface Definition Language
  See IDL
Internationalization features, POSIX, 4–17
International Organization for Standardization
  See ISO
International Telegraph and Telephone Consultative Committee
  See CCITT
Internet, 1–11, 6–7
  mail features, 5–13
Internet Engineering Task Force
  See IETF
Interoperability
  database, 7–15
  multivendor, 2–5
Interpreters, 4–5
Interprocess communication, 3–4
IPX protocols, 6–21
ISO (International Organization for Standardization), 6–11, A–1
  standards, 2–2, 6–3

J

Job scheduling, 3–3

K

KDC (Key Distribution Center), 6–8
Kernel, OpenVMS, 3–1
  threads, 3–3
Key Distribution Center
  See KDC

L

LAD (local area disk), 1–6
  protocols, 6–9
LAD Control Program (LADCP) utility, 3–14
LAN Ancilliary Control Process (LANACP), 3–21
LAN Control Program (LANCP) utility, 3–21
Languages
  ANSI/ISO, 2–4, A–2
  compilers, 4–5
LAN Manager, 6–18
LANs (local area networks), 1–11
  Ethernet, 1–11, 6–18
  Ethernet adapter, 1–6
  extended, 1–11
  FDDI, 1–11
  FDDI adapter, 1–6
  token ring, 1–11, 6–18
  token ring LAN adapter, 1–6

LAST (local area systems transport) protocols, 6–9
LAT Control Program (LATCP) utility, 3–10, 3–14
LAT software, 1–6, 3–14
Lexical functions, 5–4
Librarian utility (LIBRARIAN), 4–7
License Management Facility
  See LMF
Linker utility (linker), 4–7
LMF (License Management Facility), 3–10, 7–7
Local area disk
  See LAD
Local area network
  See LAN
Local area transport
  See LAT
Local area VMScluster environments, 1–8
Lock manager, 3–4
Log files, 3–12
Logging in
  to operating system, 5–1
  to remote system, 5–2
Logical names, 5–3
Log Manager Control Program (LMCP) utility, 3–15
LSE/SCA, 4–5, 6–16

M

MACRO
  MACRO-32 Compiler, 4–6
  MACRO-64 Assembler, 4–6
  VAX MACRO, 4–6
MACRO-32 Compiler
  See MACRO
MACRO-64 Assembler
  See MACRO
Mail, electronic, 5–13
MAILbus, 6–5
Mail utility, 5–13
Mail utility (MAIL), 6–23
MailWorks, 5–13
MailWorks for OpenVMS, 6–23
Management
  complex distributed environments, 7–9
  data, 4–19
  device, 3–11
  large data centers, 7–4
  large systems, 2–13
  network, 2–15, 3–14, 6–6
  OpenVMS Management Station, 3–15
  OpenVMS servers for PCs, 6–23
  PATHWORKS network, 6–24
  performance, 7–8
  production systems, 7–4
  remote systems, 7–9
  setting up a network node, 3–14
Management (cont'd)
  storage, 3–11, 7–7
  system, 2–1, 2–15, 3–14
  VAXclusters, 2–15
  VMSclusters, 3–14, 3–20
Management services
  middleware, 6–11
ManageWORKS, 6–24
Mathematical functions, 4–10
Memory management subsystem, 3–1
Memory modules, 1–7
Menus, DECwindows Motif, 5–5
Merge utility (MERGE), 5–11
Message Router, 6–5
Message utility (MESSAGE), 4–8, 4–17
Middleware services
  categories of services and related products, 6–11
  client/server software, 2–7, 6–10
  for distributed applications, 2–9
  multivendor integration, 1–2, 6–9
  product implementations, 6–12
  software products, 1–2, 2–6, 6–9
  support for standards, 2–6, 6–11
  use on other vendor systems, 2–9
MMS, 4–9, 6–16
Module Management System
  See MMS
Monitor utility (MONITOR), 3–12
Mount utility (MOUNT), 3–23
Mouse, 5–5
MSCP server software, 3–21
Multimedia documents
  linking data, 5–7
  preparation and control, 5–12
Multiprocessing, 1–4
  symmetric, 1–4, 3–6
Multivendor environments, 6–2
Multivendor integration, 1–2
Multivendor interoperability, 2–5

N

National character set (NCS) utility, 4–9
National Institute of Standards and Technology
  See NIST
NET$PROXY.DAT (network proxy database file), 3–18, 3–19
NetBEUI, 6–18
NETPROXY.DAT (network user authorization file), 3–18
NetWare, 6–18, 6–19
  protocols, 6–21
Network File System (NFS) for OpenVMS, 6–7
Network operating system
  See NOS

Networks, 1–4, 1–8
DECnet, 6–2
distributed processing capability, 2–10
heterogeneous, 6–2
integrated, 6–2
managing, 2–1, 6–6, 6–17
open, 2–5
OSI, 6–2
PATHWORKS connectivity, 6–20
PATHWORKS transports, 6–18
proxy database file, 3–18
remote file operations, 5–11
routing, 1–11
security, 3–17, 6–8
software, 1–2
standards, A–2
TCP/IP, 6–2, 6–7
user applications, 6–6
Network user authorization file
See NETPROXY.DAT
NFS network file system, 1–10
protocols, 1–10
server software, 2–5
support for file access, 2–10
NIST (National Institute of Standards and Technology), 2–2, A–1
NOS (network operating system), PATHWORKS services, 6–18
Notes, 5–13

O

ObjectBroker, 6–15
Object files, 4–5
Object Management Group
See OMG
Objects
object-oriented data model, 5–7
object-oriented programming, 4–5, 6–15
OLTP (online transaction-processing) applications, 2–13
OMG (Object Management Group), 6–11, 6–15, A–1
Online transaction processing
See OLTP applications
OPCOM message routing, 7–5
Open3D, 6–17
Open buses and interconnects, 1–6
Open Software Foundation
See OSF
Open specifications, 2–2
Open systems, 2–1
Open Systems Interconnection
See OSI
OpenVMS Alpha operating systems, 1–13
migrating to, 1–3
OpenVMS Alpha System-Code Debugger, 3–4, 4–8
OpenVMS Debugger (debugger), 4–7
OpenVMS Management Station, 3–15
OpenVMS systems
availability, 7–2
commercial-strength software, 7–1
compatibility, 1–3
compliance with open standards, 1–1
components, 3–1
configurations, 1–1, 1–5
debugger, 4–7
dependability, 2–11, 7–2
description, 1–1, 3–1
device drivers, 3–3
disk servers, 6–23
distributed production servers, 7–2
file servers, 6–23
growth potential, 1–15
I/O connections, 1–5
in distributed environments, 6–2
in multivendor environments, 6–2, 7–2
in open environments, 6–2
installation and configuration, 3–8
login access, 5–1
maintaining, 3–12
monitoring, 3–12
open systems capability, 2–1
portable applications, 4–3
POSIX interfaces, 3–4
print servers, 6–23
production systems, 2–11, 7–1
protection against obsolescence, 1–15
scalability, 1–15
security, 3–17
servers, 1–2, 2–7, 6–22
software, 1–1
standalone, 1–7
support for standards, 2–2, A–1
system management, 2–15, 3–7
system services, 3–4, 4–11
tuning, 3–12
user environments, 5–2
user interfaces, 5–1
utility programs, 3–5
OPS5, 4–6
OSF, 2–2, A–1
OSF/Motif standards, 2–4, 5–4
OSF AES, 2–2, 6–11, A–1
OSF Distributed Computing Environment
See DCE
OSI
layers, 6–3
OSI (Open Systems Interconnection), 1–9
model, 2–2, 2–5, 6–3

OSI Applications Kernel (OSAK), 6–4

P

Pages, 3–2
Pascal, 4–6
Passwords, 3–17
  user, 5–1
Patch utility (PATCH), 4–8
PATHWORKS
  backup, 6–24
  client/server environment, 2–7
  clients, 6–18
  environments, 6–18
  Macintosh clients, 2–7
  network connectivity, 1–6
  networking connections to Macintosh clients, 6–20
  networking connections to PC clients, 6–20
  network management, 6–24
  network transports, 1–9, 6–18, 6–21
  OpenVMS management servers, 6–23
  OpenVMS servers, 1–11, 2–7, 6–22
  PC clients, 1–11, 2–7, 6–18
  products, 6–19
  servers, 6–18
PATHWORKS V5 for OpenVMS (LAN Manager), 6–19
PCA, 6–16, 7–5
PCI (personal computer interface), 1–6
Performance
  improving I/O with DECram, 7–8
  maintaining system, 3–12
  management tools, 7–8
  monitoring and tuning, 2–14, 3–12
  virtual I/O cache, 3–13
PHIGS, 6–16
PL/I
  DEC PL/I for OpenVMS Alpha, 4–6
  VAX PL/I, 4–6
Platforms
  Alpha, 1–3
  VAX, 1–3
POLYCENTER Accounting Chargeback, 7–5
POLYCENTER Capacity Planner, 7–5, 7–9
POLYCENTER Console Manager, 2–13, 7–5, 7–10
POLYCENTER File Optimizer for OpenVMS
  See File Optimizer
POLYCENTER Hierarchical Storage Management (HSM) for OpenVMS
  file shelving, 2–14
POLYCENTER HSM for OpenVMS, 7–5
  file shelving, 7–8
POLYCENTER management products, 2–16, 6–17
POLYCENTER Performance Advisor, 7–6, 7–9
POLYCENTER Performance Data Collector, 7–9
POLYCENTER Performance Solution, 2–14, 7–8
POLYCENTER Scheduler, 2–15, 7–6, 7–9
POLYCENTER Software Distribution Manager, 2–15, 7–7
POLYCENTER Software Installation utility, 3–8, 7–7
POLYCENTER System Watchdog, 7–5, 7–9
Portability
  application, 2–3
  developing programs, 4–3
  POSIX applications, 2–4, 4–3, 4–13
  user, 2–4, 5–1
Portable applications, 3–4
  on VAX and Alpha platforms, 1–14, 4–3
POSIX
  accessing the environment, 5–2
  application development utilities, 4–15
  callable interfaces, 4–16
  commands, 5–7
  complex utilities, 4–15
  container file systems, 5–10
  ed utility, 5–12
  file commands and utilities, 5–12
  files, 5–10
  file specifications, 5–10
  in DCL environment, 4–14
  installation and management, 3–14
  interactive interface, 4–15
  internationalization tools, 4–17
  OpenVMS environment, 2–4, 3–5, 5–2
  pipes, 4–15
  portable applications, 2–4, 3–4, 4–3
  programming, 4–13, 4–15
  real-time functions, 4–16
  real-time interprocess communication, 3–4
  shell, 2–5, 4–14, 5–7
  standards, 2–3, 5–2, A–1
  support of XPG3 BASE specifications, 4–17
  support of XPG4 BASE profile specifications, 4–17
  system services, 4–15
  using ISO C language, 4–15
  using OpenVMS capabilities, 4–14
  using RMS files, 4–19
  utilities, 2–5
  vi editor, 5–12
Presentation services
  middleware, 6–11
Print operations, 3–13
Process and time management subsystem, 3–1
Processes, 3–1, 5–2
  interprocess communication, 3–4
  POSIX interprocess communication, 3–4
Processing
  centralized, 1–4
  distributed, 1–4, 2–1
  modes, 1–3

Processing (cont'd)
priorities, 3–3
real-time, 1–3
transaction, 2–13
vector, 3–6
Processors
Alpha, 1–1, 1–14
VAX, 1–1, 1–14
Production systems
dependability, 7–2
description, 7–1
managing and monitoring, 7–4
needs, 7–1
OpenVMS capabilities, 2–1, 2–11
OpenVMS management tools, 7–4
Program development, 4–1
Programming
common environments, 4–1
modular techniques, 4–2
software development tools, 4–3
to standards, 4–2
Protected subsystems, 3–19
Protocols
DNA, 1–9, 1–10, 2–10
IPX, 6–21
LAD, 3–14, 6–9
LAST, 6–9
LAT, 3–14, 6–9
NetWare, 6–21
networking, 1–8
NFS, 1–10
OSI, 1–9, 1–10, 2–10
TCP/IP, 1–10, 2–10, 6–7
two-phase commit, 3–7, 7–13
Proxy accounts, 6–8
Proxy Agent, PC NSI, 6–15

Q

Queue commands, 3–13
Queue manager, 3–21
Queues
controlling, 3–21
execution, 3–13
generic, 3–13
management commands, 3–13

R

RAID (Redundant Array of Independent Disks), 7–7
RBMS (Remote Bridge Management System), 7–7
Real-time processing, 1–3
Record Management Services
See RMS
Records, RMS, 4–19
Recoverability, 2–11
Reliability, 2–11
Reliable Transaction Router
See RTR
Remote Bridge Management System
See RBMS
Remote services, booting, 6–24
Resource Broker, 2–9, 6–14
RIGHTSLIST.DAT (user rights database file), 3–18
RISC
architecture, 1–13
RMS, 3–4, 4–2, 4–19
files and records, 4–19
routines, 4–19
utilities, 4–19
RMS Journaling, 3–23, 7–6
data integrity, 2–12, 7–4
Routines
callable system, 4–9
high-performance sort, 4–13
industry-standard, 4–9
RTL, 4–10
RPC, 2–9, 6–14
RTLs (run-time libraries), 3–4
routines, 4–10
RTR (Reliable Transaction Router), 2–13, 7–11, 7–14
Run-time libraries
See RTLs

S

Scheduling jobs, 3–3
SCS (System Communications Services), 3–21
SCSI (Small Computer System Interface), 1–6, A–2
Security
auditing subsystem, 3–17
authentication, 3–19
B1 class, 3–17
C2 class, 3–17, A–2
file protection, 3–17
intrusion database, 3–19
LAN encryption, 6–8
network, 6–8
OpenVMS services for PC clients, 6–23
OpenVMS system, 3–16, 3–17
password, 3–17
protection mask, 3–11
software, 2–14
user access, 5–1
Servers, 2–6, 6–17
disk, 6–23
file, 6–23
mail, 6–23
OpenVMS, 1–2

Servers (cont'd)
OpenVMS distributed production, 7–2
other vendor, 7–2
PATHWORKS, 6–18
print, 6–23
terminal, 6–9
transaction processing, 6–25
VMScluster, 6–18
Session Manager, DECwindows Motif, 5–5
SET HOST command, 5–2
SEVMS (Security Enhancement Service software), 3–17
Shadow sets, 2–12, 3–22
SLS (Storage Library System), 2–14, 7–5, 7–8
Small Computer System Interface
See SCSI
SMP (symmetric multiprocessing), 1–4, 3–6
Software development life cycle, 4–3
Software development tools, 4–2, 6–15
Sort/Merge utility
high-performance routines, 4–13
Sort/Merge utility (SORT/MERGE), 5–11
Standalone backup, backing up the system disk, 3–11
Standalone configurations, 1–7
computing capabilities, 6–1
Standards
ANSI, 2–2
ISO, 2–2, 6–3
middleware support for, 6–11
open, 1–1, 2–2
OpenVMS support for, A–1
OSF AES, 2–2
OSF DCE, 2–2
POSIX, 2–2, 2–3
X/Open, 2–2
Storage management, 3–11
software, 2–14
tools, 3–11, 7–7
StorageWorks RAID Array Subsystems, 3–22
StorageWorks RAID Software for OpenVMS, 2–14, 7–6, 7–7
Swapper, 3–2
Symmetric multiprocessing
See SMP
System access
controlling, 3–10
System Communications Architecture, 1–8
System Communications Services
See SCS
System configuration utilities and commands, 3–9
System Dump Analyzer (SDA) utility, 4–9
System Generation utility (SYSGEN), 3–8
System management, 2–1, 3–14
capabilities, 2–15
tasks, 3–9
utilities, 3–9
System Management utility (SYSMAN), 3–14
System routines, callable, 4–2
System services, OpenVMS, 3–4, 4–11
System tuning, 3–12
System user authorization file
See SYSUAF.DAT
SYSUAF.DAT (system user authorization file), 3–18

T

T3 technology
See DS3 communications services
Tapes, cluster-accessible, 3–22
Tape servers, VMScluster, 3–21, 6–18
TCP/IP, 1–11
layers, 6–3
network connections, 2–10
networking products, 1–9
networks, 6–7
network transport, 6–21
protocols, 1–10, 2–10, 6–7
services for OpenVMS, 1–10, 6–2, 6–7
transports, 1–10
UNIX connections, 2–5
Terminal servers
DECservers, 6–9
multiprotocol, 6–9
Test Manager, 6–16
Text Editor and Corrector
See Editors, TECO
Text Editors
See Editors
Text formatters, 5–12
Text processors, 4–4
Text retrieval facility
See VTX
Threads
DECthreads services, 4–9
kernel, 3–3
Three-dimensional graphics applications, 6–16
Timesharing, 1–4
TMSCP server software, 3–21
Token ring LAN, 1–11, 6–18
adapters, 1–6, 6–21
Transaction interfaces, responding to forms, 1–3
Transaction processing, 7–11
ACMS, 7–12
applications, 2–13, 7–10
client/server systems, 7–11
distributed environments, 4–12, 7–11
distributed management, 3–7
monitor, 2–13, 7–12
servers to desktop clients, 6–25
software, 2–13
two-phase commit protocol, 7–13
using DECdtm, 7–13

Transactions
characteristics, 7–11
distributed, 2–13, 3–7
TSM (Terminal Server Manager), 7–7
Tuning, system, 3–12
TURBOchannel I/O interconnect, 1–6
Turnkey user accounts, 5–2
Two-phase commit protocol, 3–7, 7–13

U

U.S. Government OSI Profile (GOSIP) Test Facility, 2–2
UAFs (user authorization files)
account records, 3–10
process characteristics, 5–2
use during login, 3–10
user accounts, 5–1
UICs (user identification codes), 3–18
protection mechanisms, 5–1
Uniprocessing, 1–4
User accounts, 5–1
User authorization files
See UAFs
User identification codes
See UICs
User interfaces, 1–3
accessing different environments, 5–2
graphical, 5–2, 5–4
portable, 2–4
programming tools, 4–17
responding to forms, 5–8
standards, A–2
system access, 5–1
User portability, 2–4
User rights database file
See RIGHTSLIST.DAT
Utilities
OpenVMS, 3–1, 3–5
POSIX, 4–15
Utility routines
OpenVMS, 4–13

V

VAX
architecture, 1–13
VAX APL
See APL
VAX BASIC
See BASIC
VAX C
See C
VAXcluster environments
availability, 2–11
Business Recovery Server, 7–10
client/server, 2–7
VAXcluster environments (cont'd)
configurations, 1–4, 1–7
disaster-tolerant, 1–8, 2–11
VAX DIBOL
See DIBOL
VAX DOCUMENT
See DOCUMENT
VAXELN toolkit, 4–18
VAX Environment Software Translator
See VEST
VAX MACRO
See MACRO
VAX PL/I
See PL/I
VAX platforms, 1–3
VAX processors, 1–1, 1–14
VAXsimPLUS, 7–7, 7–10
Vector processing, 3–6
Vectors, 3–6
VEST (VAX Environment Software Translator), 4–6
vi editor, 4–5, 5–12
Virtual addressing, 3–2
Virtual I/O cache, 3–13
VME bus, 1–6
VMScluster environments
alias node identifier, 3–21
availability, 7–2
batch queues, 1–8
CI, 1–8
cluster-accessible disks, 1–7
cluster-accessible tapes, 1–7
computing capabilities, 6–1
configurations, 1–4, 1–7
connection manager, 3–21
data servers, 6–18
disk servers, 3–21, 6–18
distributed file system, 3–21
distributed lock manager, 3–21
DSSI, 1–8
dual-architecture configurations, 1–7
execution queues, 3–21
generic queues, 3–21
hardware, 1–8
HSC, 1–8
interconnects, 1–8
management tools, 7–9
managing, 2–15, 3–14, 3–20
mixed-interconnect, 1–8
multisite, 1–8
PATHWORKS support for, 6–23
print queues, 1–8
queue manager, 3–21
resource access, 3–21
resource locking, 3–21
SCSI interconnect, 1–6
servers, 6–18
shared disk resources, 3–22

VMScluster environments (cont'd)
shared processing and printer resources, 3–21
software, 1–8, 3–20
software components, 3–20
System Communications Services (SCS), 3–21
tape servers, 3–21, 6–18
VMSINSTAL command procedure, 3–8
Volume shadowing, 3–22, 7–6
data availability, 2–12, 7–3
VTX, 5–13

W

WANrouter, 6–9
WANs (wide area networks), 1–11
Wide area networks
See WANs
Wildcard characters, 5–9
Windowing, 1–3, 5–5
graphical interfaces, 5–4
Workspace, DECwindows Motif, 5–5

X

X.25 software
CCITT recommendations, 6–5, A–2
packet-switching networks, 1–11, 6–5
X.400 messaging services, 6–5
X/Open consortium, 2–2, 6–11
X/Open Portability Guide Issue 3
See XPG3
X/Open Portability Guide Issue 4
See XPG4
X/Open standards, 2–2, 6–11, A–1
XPG3 (X/Open Portability Guide Issue 3), 2–2, A–1
BASE specification, 2–2, 2–3
Common Applications Environment, 4–17
XPG4 (X/Open Portability Guide Issue 4), 2–2, A–1
BASE profile specifications, 2–2, 2–3
X Window System, 5–4, A–1